



Пловдивски университет „Паисий Хилендарски”

Факултет по математика и информатика

Катедра: Софтуерни Технологии

ДИПЛОМНА РАБОТА

Тема:

**Уеб система за управление на дейности
в здравно заведение**

Дипломант: Георги Димов Димов

Научен ръководител: гл. ас. д-р Христо Христов

Специалност:

Софтуерни технологии и дизайн

фак. №: 1701681093

Съдържание

Увод	7
1. Структура на дипломната работа.....	7
2. Представяне на проблема	8
Глава 1 - Анализ на необходимостта от система за управление на здравно заведение.....	9
1. Ползи от съществуването на системата	9
2. Използвани технологии	10
2.1 Front-End технологии.....	10
2.1.1 HTML.....	10
2.1.2 CSS.....	10
2.1.3 Bootstrap	11
2.1.4 Javascript.....	12
2.2 Back-End технологии	13
2.2.1 PHP.....	13
2.3 База данни - MYSQL	14
2.4 Интегрирана среда за разработка (IDE) и допълнителни пакети.....	15
2.4.1 Sublime Text	15
2.4.2 TCPDF	15
2.4.3 WAMP Server	16
Глава 2 – Проектиране на софтуерна архитектура, потребителски интерфейс и база данни	17
1. Системни изисквания.....	17
1.1 Нефункционални	17
1.2 Функционални	18
2. Архитектура на софтуера	19
2.1 Модули	19
3. Навигационно меню	20
4. Случаи на употреба	21
5. Сценарий “Час за преглед”	22
6. Представяне на процес при регистрация	23
7. Архитектура на база данни.....	24
Глава 3. Реализация на уеб базираната система	25
1. Бизнес слой	25
1.1 Файлова структура	25
1.2 Разработване	26

2.	Слой потребителски интерфейс	27
2.1	Начална страница	27
2.2	Страница “За нас”	32
2.3	Страница “Контакти”	33
2.4	Ауторизация	34
2.4.1	Вход	34
2.4.2	Регистрация	37
2.5	Потребителски групи	41
2.5.1	Администратор	41
2.5.2	Доктор	49
2.5.3	Пациент	52
3.	Слой база данни	58
	Глава 4. Перспективи, приноси и бъдещо развитие	61
1.	Перспективи за бъдещо развитие	61
2.	Приноси	61
	Заклучение	62
	Библиография	63

Списък на използваните фигури и снимки

Фигура 1 - Архитектура на софтуера, стр.19

Фигура 2 - Диаграма на навигационно меню, стр.20

Фигура 3 – Диаграма, показваща конкретен случай на употреба, стр.21

Фигура 4 - Диаграма "час за преглед", стр.22

Фигура 5 - Диаграма “процес на регистрация”, стр.23

Фигура 6 - Диаграма “архитектура на БД”, стр.24

Снимка №1 – Конекция към база данни, стр.25

Снимка №2 – Файлова структура, стр.26

Снимка №3 – Начална страница, стр.27

Снимка №4 - Кодово представяне на слайдър, стр.28

Снимка №5 – Секция от началната страница, стр.29

Снимка №6 - Втора секция от начална страница, стр.29

Снимка №7 – Таблица с работни часове и часовник, стр.31

Снимка №8 – Интеграция на css файл, стр.31

Снимка №9 – Интеграция на Bootstrap, стр.31

Снимка №10 – Колона за контакти, стр.32

Снимка №11 – Страница “За нас”, стр.32

Снимка №12 – Локация, стр.32

Снимка №13 – Страница “Контакти”, стр.33

Снимка №14 – Формуляр за изпращане на писмо до администрация, стр.34

Снимка №15 – Страница за вход в системата, стр.34

Снимка №16 – Вход за админ, стр.35

Снимка №17 – Формуляр за вход в системата, стр.35

Снимка №18 – РНР код за реализиране на вход, стр.36

Снимка №19 – Страница за регистрация, стр.37

Снимка №20 – Формуляр за регистрация, стр.39

Снимка №21 – РНР код за реализиране на регистрацията, стр.40

Снимка №22 – Екран на табло за администратор, стр.41

Снимка №23 – Екран за добяване на лекар, стр.42

Снимка №24 – Екран за добяване на пациент, стр.42

Снимка №25 – Формуляр за добавяне на лекар, стр.43

Снимка №26 – РНР код за реализиране на добавяне на лекар, стр.44

Снимка №27 – Екран за изтриване на пациент, стр.44

Снимка №28 – Формуляр за изтриване на пациент, стр.45

Снимка №29 – Екран за търсене на лекар, стр. 45

Снимка №30 – Формуляр за търсене на лекар,стр.46

Снимка №31 – Формуляр със списък с минали прегледи, стр.48

Снимка №32 – Екран на табло за лекар, стр.49

Снимка №33 – Екран със списък с насрочени часове, стр.49

Снимка №34 – Екран за предписване на бележки, стр.50

Снимка №35 – Екран със списък на отминали прегледи, стр.50

Снимка №36 – Екран за промяна на потребителски данни, стр.51

Снимка №37 – Екран за записване на час за преглед, стр.52

Снимка №38 – Формуляр за записване на час за преглед, стр.53

Снимка №39 - Екран със списък на прегледи, стр.54

Снимка №40 - РНР условен оператор за отказване на преглед, стр.54

Снимка №41 - Екран със списък на отминали прегледи, стр.54

Снимка №42 - Екран с предписана от лекар бележка, стр.55

Снимка №43 - Автоматичен генератор на PDF документи, стр.56

Снимка №44 - Екран за промяна на потребителски данни, стр.56

Снимка №45 - Формуляр и РНР код за промяна на потребителски данни, стр.57

Снимка №46 - Структура на таблица “пациенти”, стр.59

Снимка №47 - Структура на таблица “администратор”, стр.59

Списък на съкращения

Съкращение	Пълно наименование (EN)	Пълно наименование (БГ)
HTML	HyperText Markup Language	Език за маркиране на хипертекст
CSS	Cascading Style Sheets	Каскадни таблици със стилове
PHP	Hypertext Preprocessor	Хипертекстов препроцесор
XML	Extensible Markup Language	Разширяем език за маркиране
HTTP	Hypertext Transfer Protocol	Протокол за пренос на хипертекст
URL	Uniform Resource Locator	Унифициран локатор на ресурси
W3C	The World Wide Web Consortium	Консорциумът за световната мрежа
AJAX	Asynchronous JavaScript and XML	Асинхронни JavaScript и XML
CMS	Content Management System	Система за управление на съдържанието
IIS	Internet Information Services	Интернет информационни услуги
SQL	Structured Query Language	Език за структурирани заявки
IDE	Integrated development environment	Интегрирана среда за разработка
API	Application Programming Interface	Интерфейс за приложно програмиране

Увод

В настоящата работа е представена реализацията на проект, който описва идея за управление и организация на здравно заведение. В дипломната работа се разгледани въпроси за управлението на дейности в здравно заведение чрез софтуерна система. В работата е представен процесът на събиране на изисквания, проектиране на модел на архитектура, дизайн, база данни и др. Представени са типичните дейности в едно болнично заведение, които са описани чрез случаи на употреба и сценарии. Разгледани са подробно етапите на проектиране и програмиране на сценариите, както и въпроси свързани с внедряването в реална среда на завършена версия на приложението.

Основна цел на дипломната работа е разработването на уеб система за управление на дейности в здравно заведение.

За постигане на основната цел са формулирани следните **задачи**:

1. Да се проучат уеб системите за администриране и управление на болнични заведения;
2. Да се опишат основни типови дейности за здравно заведение в едно с ролите и отговорностите на актьорите, които ги извършват;
3. Да се подберат технологии и инструменти за разработване на софтуерната система;
4. Да се проектират модели на архитектура и база с данни на приложението;
5. Да се програмират проектираните случаи на употреба и сценарии;
6. Да се оптимизира работната версия на приложението;

1. Структура на дипломната работа

Дипломната работа се състои от увод, четири глави, списък на фигури, заключение и библиография.

В първа глава е направен обзор на технологиите за създаване на уеб базираната система. Детайлно са разгледани технологиите като Езикът за маркиране на хипертекст (HyperText Markup Language или накратко HTML), Каскадни таблици със стилове (Cascading Style Sheets или накратко CSS) , програмният език JavaScript, работна рамка Bootstrap, програмният език Хипертекстов препроцесор (Hypertext Preprocessor или накратко PHP), използвана среда за разработка и допълнителни пакети.

Във втора глава се разглежда проектирането на системата, представена е визуализация чрез различни видове диаграми.

В трета глава подробно е описано разработването на уеб базираната система за болнично заведение. Представени са най-важните екрани, визуализиращи основните функционалности на системата. Включен е част от изходния код на системата.

В четвърта глава са представени перспективи за бъдещо развитие на системата и приноси.

В заключението се обобщават постигането на основната цел, разрешаването на поставените задачи, като се анализират постигнатите резултати и се прави оценка на следващите стъпки за развитие на нова функционалност. Дипломната работа завършва с представена библиография и използвани източници в процеса на разработка на софтуера.

2. Представяне на проблема

Конкретният повод, който ме доведе до идеята за разработване на софтуерния продукт, представен в настоящата дипломна работа, е породен от създалата се епидемична обстановка и по-конкретно струпването на едно място на големи групи от хора и чакането на големи опашки, което е в разрез с противоепидемичните мерки, а също така е и времеемко. Поради тази причина, с цел подобряване на организацията пред лекарските кабинети се убедих в необходимостта и ползата от настоящата уеб-базирана система.

Днешната уеб базирана технология предлага множество онлайн услуги в почти всяка област. Всяка голяма индустрия се адаптира към дигиталния свят за свои основни и важни операции. Компютърните технологии имат потенциал за повишаване на качеството и ефективността на работния процес. Бумът на дигиталния пазар е огромен, понастоящем информационният поток е изключително бърз и динамичен. Онлайн свързаността вече е задължителна за всички добре организирани и управлявани звена. Едно такова звено е здравеопазването, където цифровизацията на информацията е хубаво да се случи бързо и ефективно. Тъй като все повече институции прибягват до помощта на дигиталния свят, изглежда удачно и извършването на определени функции в болничните заведения да се насочи в тази посока.

Здравето е най-важното нещо за всеки един човек. Взимайки за пример следния казус: Ако някой е болен или просто иска да посети лекар за преглед, той трябва да посети болницата и да чака докато лекарят е на разположение. Пациентът също така чака на опашка, докато получи извикване/назначаване на час. Ако лекарят отмени извикването поради неотложни или спешни причини, тогава пациентът попада в неприятна ситуация, в която е загубил време. Особено сега в условията на пандемия. Намираме се в критичен момент от битката срещу пандемията, момент на трескава надпревара между ваксините и мутациите на вируса. Вследствие на това, че технологиите се развиват с такива темпове, човек може да използва своя личен компютър или мобилно устройство за преодоляване на подобни проблеми и неудобства. За улеснение на пациентите и обслужващите ги лекари, следва да бъде създадена централизирана онлайн система, която максимално да спомогне за улеснение на графика.

Глава 1 - Анализ на необходимостта от система за управление на здравно заведение

1. Ползи от съществуването на системата

В настоящата точка са представени основните ползи от съществуването на уеб-системата.

Главният фокус пада върху:

- Компютризиране всички данни за пациента и за лекарите, водещо до по-голямо удобство.
- Планиране на назначаването на прегледи от пациент скоростно и безпроблемно.
- Редуциране на хартиените носители – Както информацията за пациентите, така и предписани бележки в болниците се запазват на хартия. Софтуерът използва база данни, така че данните ще се съхраняват по по-организиран начин, което ще доведе до намалянето на хартиената работа. Базата данни може да съхранява милиони записи по лесен и структуриран начин.
- Скоростно бързодействие при търсене – Посредством софтуера може да бъдат лесно търсени конкретни данни. Например, ако администратор или доктор иска да потърси информация за даден пациент, той може лесно да направи това чрез въвеждане на критерий, например име на пациент или телефонен номер, вместо тази конкретна информация да бъде търсена сред стотици или хиляди досиета на пациенти.
- Спестяване на време – Както това на пациентите, също и ще улесни и систематизира работата на лекарския персонал относно ежедневни медицински задачи.

2. Използвани технологии

В тази точка ще бъдат разгледани използваните технологии за изграждането на уеб базираната система за болнично управление. Детайлно са представени спецификите на базата данни, програмните езици за реализация на потребителския интерфейс и технологиите за създаване на логиката на системата.

2.1 Front-End технологии

В частта, обхващаща front-end, се включва обзор на използваните технологии за реализация на потребителския интерфейс на системата.

2.1.1 HTML

Езикът за форматиране на хипертекст (HyperText Markup Language) или накратко HTML, е универсалният език за описание на уеб страници. Описанието на документа става чрез специални елементи, наречени HTML елементи или маркери, които се състоят от етикети или тагове (HTML tags) и ъглови скоби (като например елемента `<html>`). HTML елементите са основната градивна единица на уеб страниците. Чрез тях се оформят отделните части от текста на една уеб страница, като заглавия, цитати, раздели, хипертекстови препратки и т.н. Най-често HTML елементите са групирани по двойки `<h1>` и `</h1>`. Клиентските браузъри го използват, за да разберат как точно да поднесат получените данни на потребителя.

HTML ни помага да зададем основната структура на дадена уеб страница. Също така е полезен за да редактираме параграфи, заглави и други основни елементи на една уеб страница. Когато някой кликне на линк в страница или въведе нов URL в адресното поле, браузърът изпраща заявка за документ от Web server-а, където е качена страницата, посредством Hypertext Transport Protocol (HTTP). Сървърът след това изпраща документа обратно до потребителя, като се визуализира на дисплея от браузъра. Всички неща, които са на този документ – текст, снимки, аудио или видео файлове – всички те са подредени посредством HTML структура.

В представената дипломна работа, HTML се използва за придаване на структура на отделните уеб страници в сайта.

2.1.2 CSS

CSS (Cascading Style Sheets) е език за описание на стилове - използва се основно за описване на представянето на документ, написан на език за маркиране. Най-често се използва заедно с HTML, но може да се приложи върху произволен XML документ. Официално спецификацията на CSS се поддържа от W3C (World Wide Web Consortium).

Правилото за писане на CSS има два основни компонента: селектор и една или повече декларации. Селекторът обикновено е HTML елемент, на който се придава даден стил. Всяка от посочените декларации съдържа атрибут и стойност на атрибута

Може да се използва по няколко начина:

- Външен стил - Най-често използван и най-близо до причината, заради която CSS е създаден е когато прилагаме външен стил (External Style Sheet) с помощта на който, контролираме множество HTML страници, като нужните параметри са зададени във външен CSS файл.
Бихме могли да зададем и използваме дори CSS файл, който не се хоства на сървър, което в някои случаи оптимизира ресурсите и прави зареждането на страниците по-бързо за потребителя.
- Вътрешен стил - Друг начин за приложение е т.н. вътрешен стил – Internal Style Sheet), който се използва за да се зададе вид на един конкретен HTML документ, като свойствата се задават с помощта на специален таг style в секцията head на HTML документа.
- Вграден стил - Друг начин за прилагане на CSS е т.н. Inline Styles – вътрешни за HTML таговете стилове. Тогава, добавяме CSS стилове, като атрибут директно в HTML тага.

В дипломната работа, CSS се използва за създаване на свойства на отделните HTML елементи.

2.1.3 Bootstrap

Bootstrap е платформа която е комбинация от HTML, CSS и JavaScript код създадени от екипа на Twitter. Напълно поддържа CSS3 и HTML5. Това е важно тъй като CSS3 и HTML5 са се превърнали в стандарт за уеб дизайна през последните години. Съвместимостта с всички основни браузъри го прави предпочитан от голяма част от програмистите. Сайтовете изградени с Bootstrap изглеждат зашеметяващо и работят правилно независимо от устройството и/или браузърът който посетителите използват. С помощта на комбинация от Javascript, CSS и интелигентни мрежи, Bootstrap позволява да се изпълни почти всеки елемент от дизайна който може да се изиска.

Предимствата на този фреймърк са, че е:

- Бърз;
- Напълно приспособим;
- Лесен елегантен и интуитивен;
- Отзивчив от самото начало;
- Притежава перфектна grid система;
- Обширен списък на компонентите;
- Пакетни JavaScript плъгини;
- Притежава всички елементи необходими за разработката на уеб сайт – таблици, форми, бутони падащи менюта и т.н;
- Предоставя стилове за всички основни HTML елементи;

Bootstrap екипът постоянно актуализира рамката. Всички изброени по-горе характеристики означават много добре проектирани шаблони. Bootstrap шаблоните са лесни за поддръжка и не е необходимо да бъдеш уеб разработчик за да управляваш своя уеб сайт.

В дипломната работа, помощта на работната рамка Bootstrap се изразява в интегриране на интуитивен дизайн.

2.1.4 Javascript

JavaScript представлява интерпретируем език за програмиране от високо ниво, разпространяван с повечето модерни Уеб браузъри.

Като език с множество парадигми, JavaScript поддържа стилове за програмиране, ориентирани към събития, функционални и императивни (включително обектно-ориентирани и прототипни). Наред с HTML и CSS, JavaScript е една от основните технологии в World Wide Web. JavaScript позволява създаването на интерактивни уеб страници и е съществена част от модерните уеб приложения. По-голямата част от уебсайтовете го използват, като големите и модерни уеб браузъри имат специален JavaScript механизъм, който да го изпълни.

Възможности на JavaScript:

- Зареждане на данни чрез AJAX;
- Ефекти с изображения и HTML елементи: скриване/показване, пренареждане, влачене, слайд шоу, анимация и много други;
- Управление на прозорци и рамки;
- Разпознаване на възможностите на браузъра;
- Използване на камера и микрофон;
- Създаване на 3D графики WebGL;
- По-добър и гъвкав потребителски интерфейс;

В уеб системата, посредством JavaScript е изграден часовник, отразяващ часа в реално време. Горепосоченият скриптов език е използван и за изписване на валидационни грешки, както и за съобщения за успешно извършена операция.

2.2 Back-End технологии

За back-end на системата е използван езикът PHP.

2.2.1 PHP

PHP е скриптов език за програмиране, който е проектиран за създаване на динамични интернет страници. Предимно се използва за разработване на уеб сайтове, като PHP библиотеките значително улесняват този процес, тъй като спестяват писането на сложен и обемен програмен код.

Той е един от най-популярните езици за програмиране в Интернет, с който са изградени едни от най-използваните CMS системи (системи за управление на съдържанието) по света. Мнозина експерти го определят, като изключително подходящ за абсолютно начинаещи или едва прохождащи хора, които искат да програмират, като в същото време предлага разширени възможности и среда за професионално програмиране.

PHP може да бъде използван от всички най-големи операционни системи, включително Windows, Linux, много UNIX варианти, като OpenBSD, Solaris, HP-UX и др., MacOS can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, Mac OS X и много други. PHP се поддържа от повечето уеб сървъри в наши дни, като най-популярните Apache, IIS и много други.

Сред предимства на езика са:

- PHP е open source и разработването на софтуер е безплатно и бързо;
- Има множество работни рамки с голям набор от лесно достъпни инструменти;
- PHP е platform independent, което означава, че приложенията, писани на езика, се поддържат от всякакви операционни системи;
- Може да се интегрира с други програми езици;
- Лесен за научаване;
- Разполага с удобен начин за свързване с бази от данни, като за целта разполага с built-in модул;
- Интегрира се с множество видове системи за менажиране на бази от данни;
- Лесна поддръжка;
- Online support и голямо community;
- Бързо и ефективно тестване;
- Сигурност;
- Стабилност;

В дипломната работа, езикът PHP е използван за изграждането на логиката на системата, посредством логически и циклични структури, характерни за езика.

2.3 База данни - MYSQL

Бази данни наричаме структурирано количество от данни. Бази данни представляват всичко от най-обикновен списък за пазаруване, през изображения в галерия до огромните количества информация в корпоративните мрежи.

MySQL е най-популярната система за управление на SQL бази данни с отворен код. Тя се разпространява и поддържа от Oracle Corporation. SQL е съкращение от Structured Query Language (Структуриран език за заявки), предназначен за създаване, обработка и четене на бази данни, които представляват пакети от свързана информация, съхранявана в таблици.

MySQL базите данни са релационни. Релационните бази данни съхраняват данни в отделни таблици, вместо да поставят всичките данни в една голяма директория. За по-бърза работа структурите от бази данни са организирани във физически файлове. Логическият модел, с обекти като бази данни, таблици, изгледи, редове и колони предлага гъвкава програмна среда. Вие настройвате правила, управлявате връзките между различните полета с данни, като „one-to-one“, „one-to-many“, „unique“, „required“ или „optional“, както и различни показалци между различните таблици. Базите данни налагат тези правила, така че с добре-проектирана база данни разработваните приложения никога няма да се сблъскат с несъвместими, дублиращи се, остарели или липсващи данни. MySQL е софтуер с отворен код. „Отворен код“ означава, че е възможно за всеки да модифицира софтуера. Всеки желаещ може да свали MySQL софтуера от Интернет и да го използва, без да заплаща нищо. Ако искате може да изучите изходния код и да го промените така, че да отговаря на вашите нужди. Разбира се, ако възнамерявате да използвате MySQL за комерсиални цели трябва да се сдобие и със съответния лиценз.

Основни предимства за използването на MySQL са:

- Сигурност на данните;
- Висока производителност;
- Бързодействие;
- Цялостна транзакционна поддръжка;
- Пълен контрол на работния поток;
- Гъвкавостта на отворения код;
- Инструменти за бекъп;

В дипломната работа посредством MySQL е изградена база данни, съдържаща таблици, където се съхранява цялата информация за потребителите на сайта.

2.4 Интегрирана среда за разработка (IDE) и допълнителни пакети

2.4.1 Sublime Text

Използваната среда за разработка при изграждането на системата е Sublime Text.

Sublime Text е междуплатформен софтуер за обработка на изходен код и текстов редактор с приложно-програмен интерфейс (API) написан на Python. Функционалността му може да бъде допълвана от потребителите чрез плъгини. Повечето от допълненията са с лиценз за свободен софтуер и се разработват и поддържат от потребителите.

Характеристики и предимства:

- Автодовършване, подчертаване на синтаксиса;
- Персонализиране;
- Лек, бърз и стабилен;
- Добра търсачка;
- Екрани, панели;

2.4.2 TCPDF

Един допълнителен пакет, използван в представената дипломна работа, е TCPDF.

TCPDF е безплатен PHP клас с отворен код за генериране на PDF документи. TCPDF е единствената PHP-базирана библиотека, която включва пълна поддръжка за UTF-8 Unicode. TCPDF е и една от най-използваните PHP библиотеки в света, тъй като вече е включена в най-популярните базирани на PHP CMS и приложения, включително: Joomla! 1.5, Drupal, Moodle, phpMyAdmin, Xoops, Elxis CMS, ImpressCMS, Jelix, SugarCRM, Symfony, TYPO3, Vtiger CRM, Yii Framework, CMS Made Simple, DaDaBIK и много други.

Някои свойства:

- Не са необходими външни библиотеки за основните функции;
- Поддържа JPEG, PNG и SVG изображения;
- Автоматично управление на горния и долния колонтитул на страницата;
- Криптиране на документи до 256 бита и удостоверяване на цифров подпис;
- PDF анотации, включително хипервръзки, прикачени файлове и файлове;
- Автоматично номериране на страници и групи страници; и други;

2.4.3 WAMP Server

В дипломната работа е използван локален сървър WAMP Server.

WAMP Server ни позволява да стартираме сървър на локалната ни машина. Традиционно един уеб сайт трябва да бъде хостван на уеб сървър, за да може да бъде достъпен. WAMP позволява на потребителят да стартира този сървър локално. Без да закупуваме домейн в интернет, можем да управляваме работата си по даден проект. С WAMP може да тестваме проекта си, докато го изграждаме, без допълнителни разходи.

След направения обзор на използваните технологии, може да се обобщи, че гореспоменатите са достатъчно надежни и спазващи всички тенденции и потребности, предвидени за разработката на софтуерния продукт.

Глава 2 – Проектиране на софтуерна архитектура, потребителски интерфейс и база данни

1. Системни изисквания

Изискванията са категоризирани в две групи – нефункционални функционални изисквания.

1.1 Нефункционални

Нефункционалните изисквания са:

- **Изисквания за използваемост** – Единственото изискване е да разполагаме с компютър или мобилно устройство с интернет връзка. Софтуерът осигурява на потребителите удобството да преглеждат своите данни дори и когато не са на територията на здравното заведение. Достъпността до информацията и използваемостта на системата е лесна. С няколко кликания потребителят може да достигне до желаната информация.
- **Изисквания за приятен и удобен потребителски интерфейс (или UI / UX дизайн)** - това несъмнено е един от основните приоритети за такава уеб система като се има предвид големият брой потенциални потребители. Както административният и медицинският персонал, така и пациентите се нуждаят от информативна и удобна система, която е лесна за ефективно използване, независимо от възрастта и компютърните умения.
- **Изисквания за производителност** – Производителността на системата ще бъде ключова, за да може всички задачи ще бъдат изпълнени в рамките на кратък период от време.
- **Изисквания за сигурност:** Софтуерът изисква идентификационна система, която ще е надеждна и ще съхранява данните.
- **Изисквания за надеждност:** Системата трябва да бъде достъпна по всяко време.

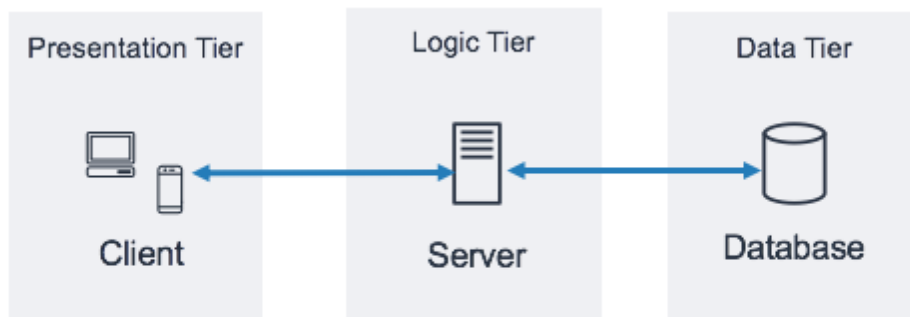
1.2 Функционални

Функционалните изисквания са:

- **Изисквания за регистрация:** Процесът на регистрация позволява на пациентите на болницата безплатна регистрация. Системата предоставя уникален идентификатор за всеки един пациент и след това добавя го добавя като запис в базата данни.
- **Изисквания за база данни:** Неизменна част от системата, разполагаща с цялата информация относно пациенти, доктори, администрация, история на прегледите, имена, имейли, телефони и т.н.
- **Изисквания за администрацията**
 1. Достъп до списък на лекари и пациенти
 2. Възможност за добавяне и изтриване на доктори и пациенти
 3. Възможност за преглед на детайли на прегледи.
 4. Възможност за преглед на отзиви от нерегистрирани потребители
- **Изисквания за лекари**
 1. Възможност за преглед на записани прегледи от стои пациенти
 2. Възможност за предписване на бележки
 3. Възможност за завършване и отлагане на преглед
 4. Възможност за редакция на част от профилна информация
 5. Възможност за преглед на списък с предписани бележки
- **Изисквания за пациенти**
 1. Възможност за записване на час за преглед
 2. Възможност за преглед на история от свои прегледи
 3. Възможност за преглеждане и принтиране на бележки от лекаря
 4. Възможност за редакция на част от профилна информация

2. Архитектура на софтуера

Използваната архитектура при разработката на софтуера е трислойна клиент/сървър архитектура. Трислойната архитектура е развита основно при работата с бази данни. Този вид архитектура се състои логически от три части – потребителски интерфейс, програмна логика и база от данни. На фигура 1 е представена нагледно архитектурата.



Фигура 1 - Архитектура на софтуера

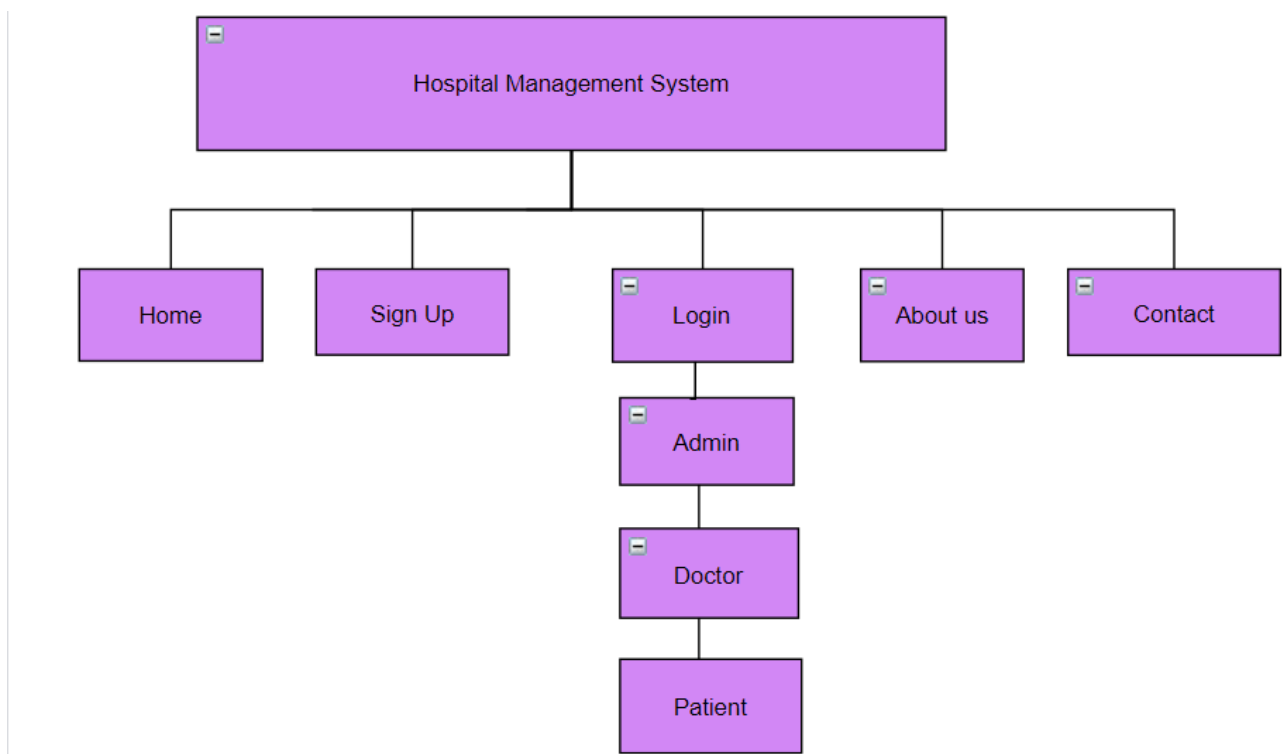
2.1 Модули

Уеб системата се състои от следните три модула:

- **Презентационен модул (Потребителски интерфейс)** - Презентационният модул е на най-високо ниво в приложението и потребителят има директен достъп до него. Освен, че служи комуникира с останалите слоеве, презентационният слой предоставя различни видове информация на потребителя.
- **Модул за бизнес логика (Междинен слой, програмна логика, сървърно-софтуерен код)** - Този слой е изтеглен от презентационния слой, и като отделен такъв, контролира функционалността на приложението като извършва различни процеси по обработката на данните.
- **Модул за данните (Конекция към база данни)** - Този модул се състои от сървър база данни. Тук информацията се съхранява и чете. В слоя за бази данни информацията се съхранява независима от бизнес логиката или сървърът за приложения. Когато данните се съхраняват в отделен слой се увеличава мащабируемостта и се подобрява производителността.

3. Навигационно меню

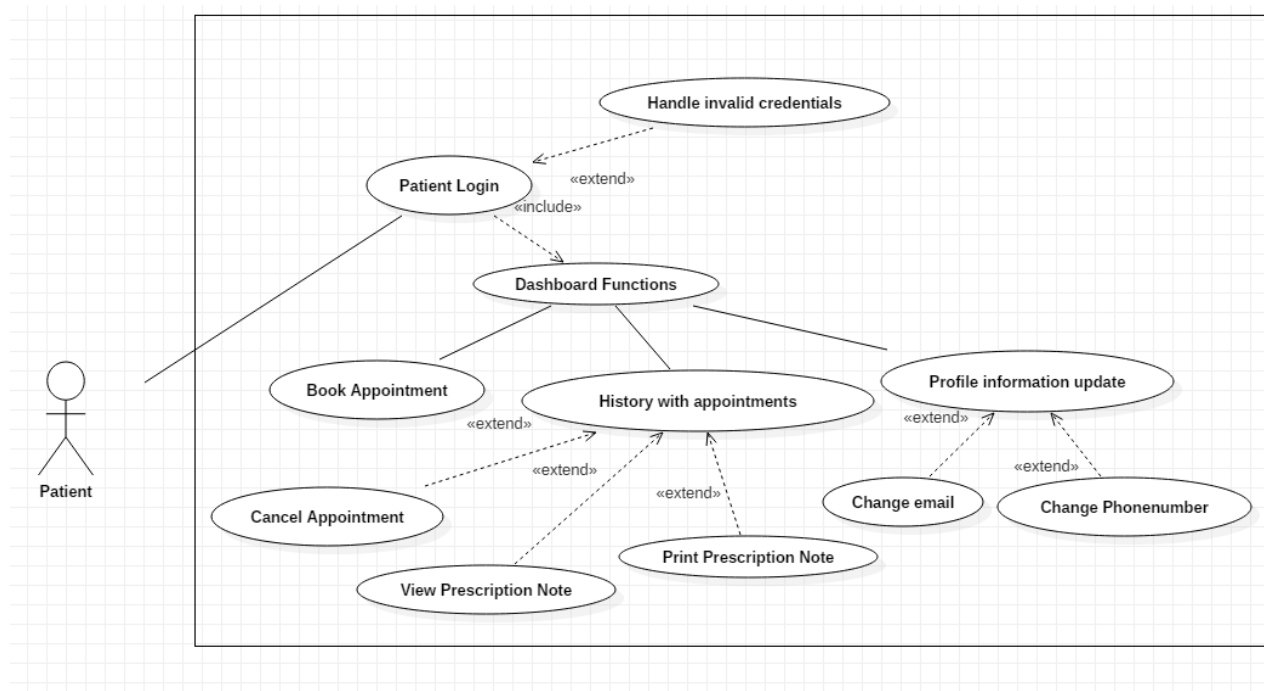
Едно от първите неща в процеса на проектиране е леснодостъпното и дружелюбно навигационно меню. Представена е визуализация чрез Hierarchy Chart диаграма на въпросния компонент (фигура 2). Диаграмата визуализира концепцията за навигационната лента на сайта. Разполагаме с 4 стандартни менюта и едно падащо меню. От навигационното меню получаваме достъп до форма за регистрация, вход в системата, страница “За нас” и страница “Контакти”.



Фигура 2 - Диаграма на навигационно меню

4. Случаи на употреба

Случаите на употреба описват начините, по които актьорите използват системата. Всеки случай на употреба описва различен сценарий или поредица от действия, които системата извършва при взаимодействие с актьорите за постигане на определен резултат или цел. В настоящата дипломна работа съществуват няколко случая на употреба. Един от най-често използваните такива е този на взаимодействие на пациент със системата. Визуализация на функциите на пациент е представена посредством Use Case диаграма на фигура 3.



Фигура 3 - Диаграма, показваща конкретен случай на употреба

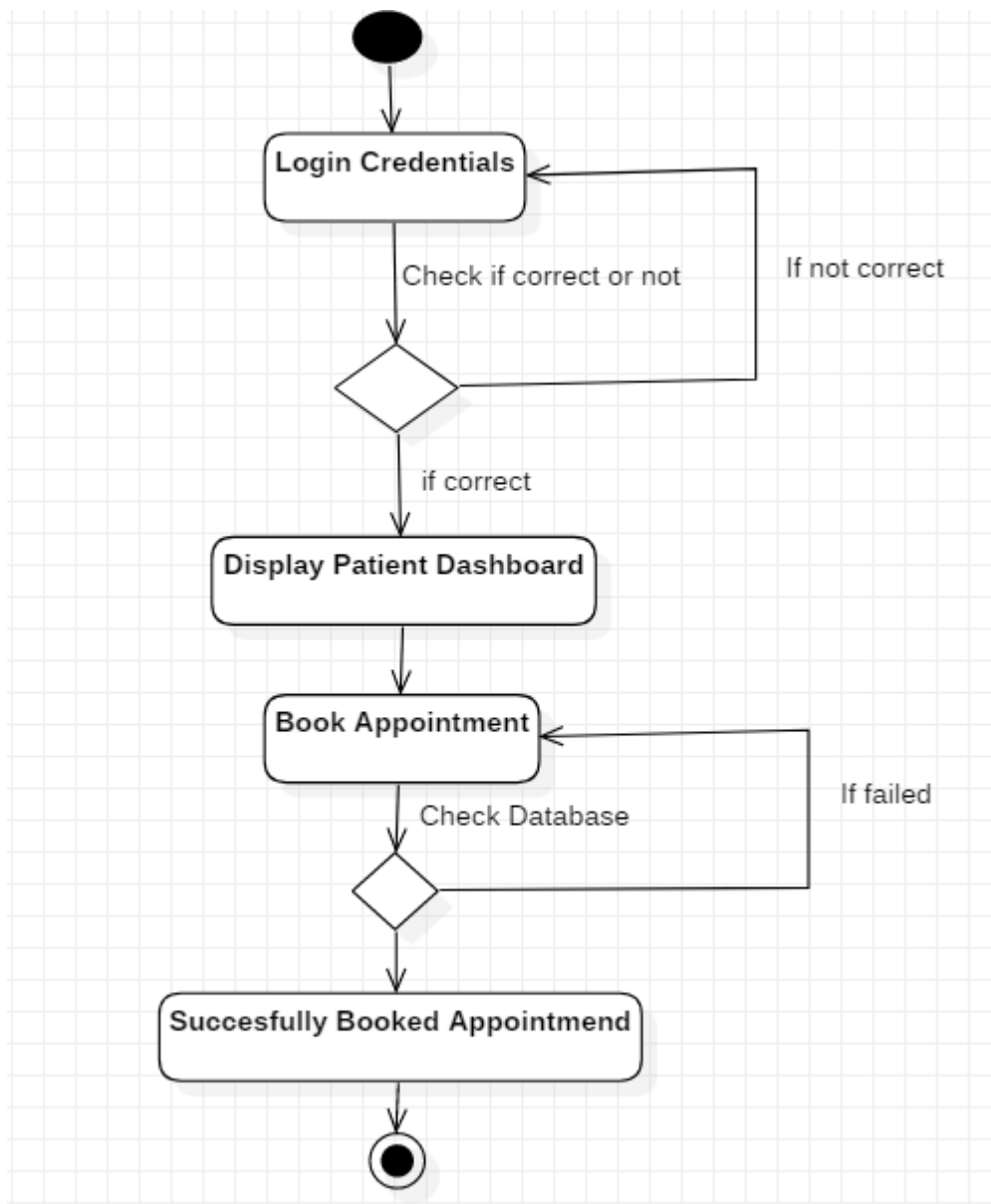
Описание на актьор и възможните операции:

Актьор на системата е всичко от външния свят, което си взаимодейства с нея. В конкретно представения случай това е пациентът. Неговите възможни функции съгласно горепосочената диаграма са както следва

- Patient Login – ауторизация в системата
 - Dashboard Function – списък с функционалности за конкретния потребител (в случая пациент)
 - Book Appointment – запис на час за преглед
 - History with Appointments – списък с минали и предстоящи прегледи
 - Profile Information Update – Актуализиране на информация за профила
 - View and Print Prescription note – Преглед и възможност за преглед на медицинска бележка предписана от лекар.
- 3.Проектиране на процес при регистрация.

5. Сценарий “Час за преглед”

В сценариите се описва последователността от действия за определено функционално изискване. В дипломната работа разполагаме с няколко различни сценария. Един от най съществените е този за записване на час за преглед. Избран е подход за илюстриране чрез Activity диграма на фигура 4.



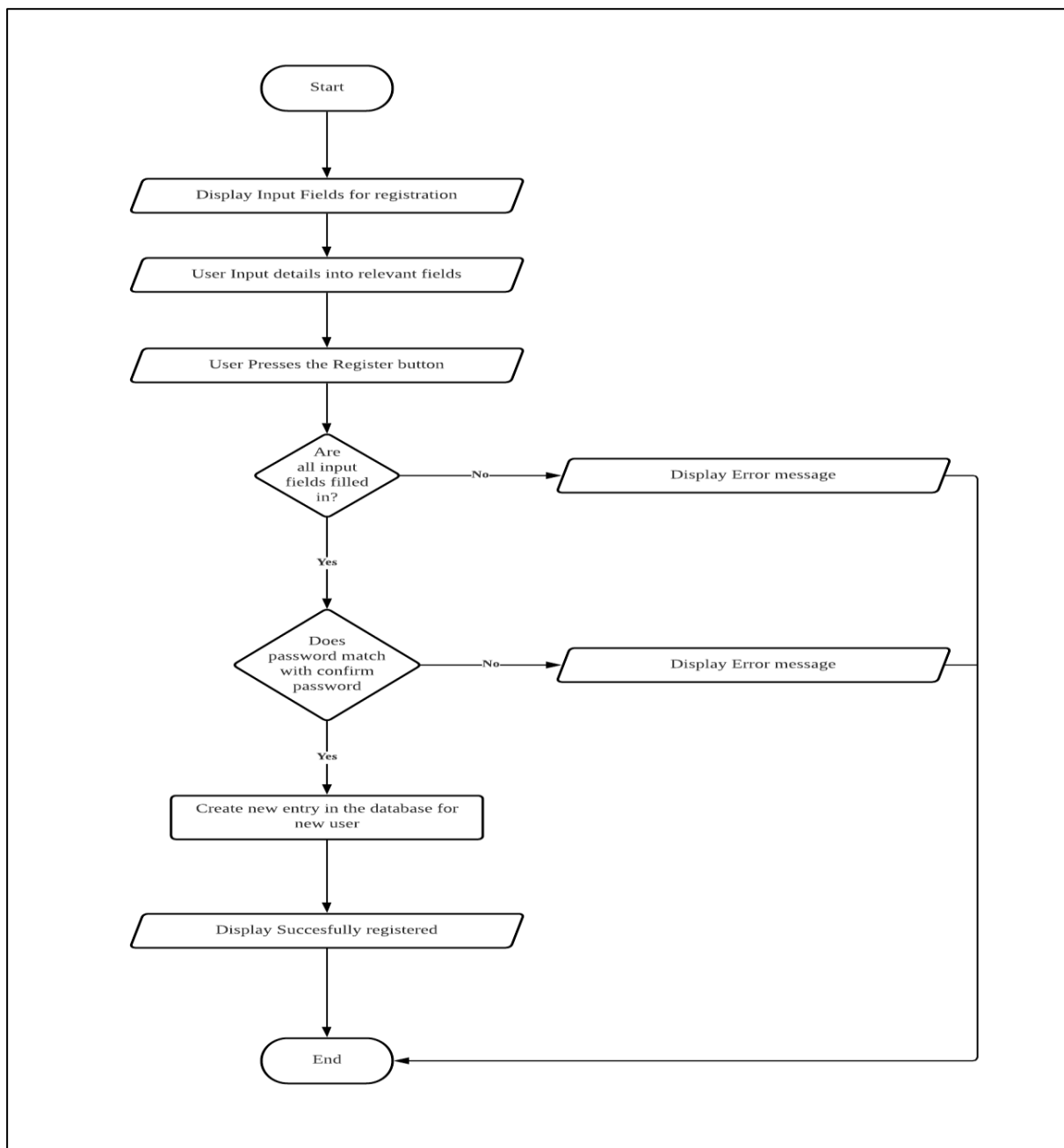
Фигура 4 - Диаграма "час за преглед"

Този вид диаграми показват конкретни дейности от процесите, които протичат в системата. Диаграма визуализира процеса на записване на час за преглед от пациента. Първоначалното изискване е потребителят да се ауторизира в системата. В случай, че въведените входни данни не съответстват на запис от базата данни, на потребителя ще бъде изписано съобщение за грешка. Ако данните съответстват със съществуващ запис, потребителят бива препращан към своето табло. Оттам той избира менюто за запис на часове за преглед. Попълва полетата, необходими за гореспоменатата операция.

Правим проверка дали всичко е въведено коректно, ако не е – потребителят ще получи съобщение за грешка, а ако всички данни са коректно избрани – ще бъде изписано съобщение за успешно записан час.

6. Представяне на процес при регистрация

В настоящата дипломна работа се наблюдава множество от процеси. Един от най използваните процеси от потребителите на системата е този за регистрация. Той следва да бъде онагледен чрез Flow Chart диаграма, илюстрирана на фигура 5.



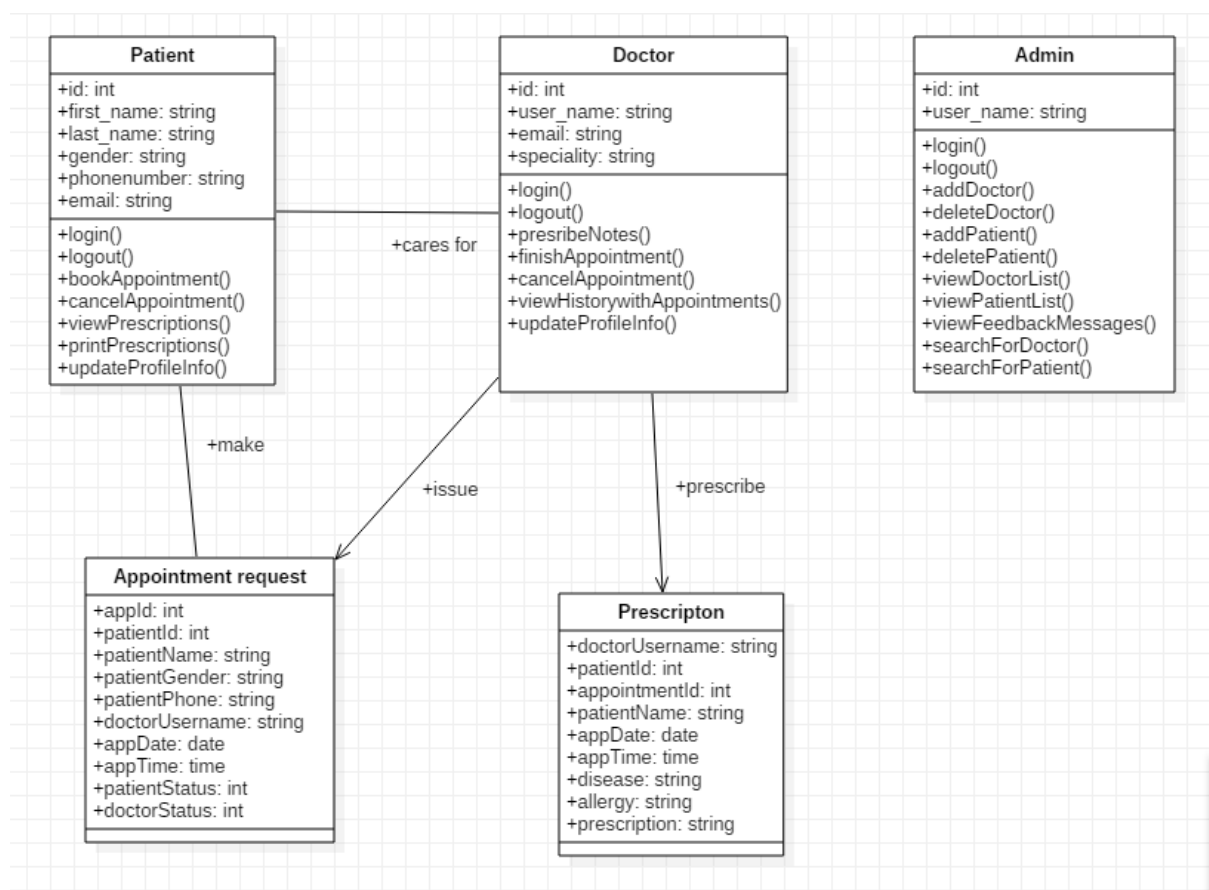
Фигура 5 – Диаграма “процес на регистрация”

Диаграмата описва процеса за регистрация на нов пациент. Първо потребителят трябва да попълни всички полета, необходими за създаване на акаунт. При натискане на

бутона за регистрация, в случай, че имаме некоректно въведени данни, на потребителя ще бъде върнато съобщение за грешка. Ако данните са били коректно въведени, се създава нов запис в базата данни и на клиента се изписва съобщение за успешно направена регистрация.

7. Архитектура на база данни

Показана е архитектурата на базата данни, визуализирането става чрез Class диаграма на фигура 6. Такъв вид диаграми показват обектните класове в системата и връзките между тях.



Фигура 6 - Диаграма "архитектура на БД"

Отделните потребители на системата са представени като индивидуални класове, характеризирайки се със своите потребителски атрибути и функции.

Глава 3. Реализация на уеб базираната система

1. Бизнес слой

Бизнес слоя съдържа компонентите, които имплементират бизнес логиката и дефинират бизнес обектите, използвани от бизнес логиката. Той обединява всички елементи, налични в проекта. Този слой е най-малко преносимият в системата. Причината е, че обикновено той е силно свързан със горния и долния слой. Той играе ролята на преход между презентационния слой и слоя на данните, затова се състои от процеси и операции. Бизнес слойът се грижи за правилната работа с много потребители едновременно. Той комуникира с базата данни за да съхранява и обработва данните в нея.

1.1 Файлова структура

В главната (основната) директория на уеб-системата разполагаме с PHP файлове за началната страница и папки за отделните потребители и функционалности. На снимка №2 е представена нагледно файловата структура на системата.

assets – в тази папка могат да бъдат намерени публични изображения, css (по принцип и js) на приложението.

db – в тази папка има два файла, единият от които е PHP файл (config.php) и служи за свързване на базата с уеб-сайта. Другият файл представлява MySQL структурата на базата данни.

Свързването на базата става посредством следния PHP код

```
<?php
//Database establish
define('DB_SERVER', 'localhost');
define('DB_USER', 'root');
define('DB_PASS', '');
define('DB_NAME', 'myhmsdb');
$con = mysqli_connect(DB_SERVER, DB_USER, DB_PASS, DB_NAME);
// Check connection
if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
```

Снимка №1 – Конекция към база данни

admin – В тази папка се намират всички файлове, описващи функционалността и привилегиите на администратора на сайта.

doctor – В тази папка са всички файлове, отговарящи за дейностите на лекарите.

patient – В тази папка могат да бъдат открити всички файлове, отговорни за осъществяването на дейностите на пациентите на системата

TCPDF – Автоматичен генератор на PDF документи

Име	Дата на промяна	Тип	Размер
admin	15.5.2021 г. 14:54	Папка с файлове	
assets	24.4.2021 г. 14:16	Папка с файлове	
db	24.4.2021 г. 12:27	Папка с файлове	
doctor	17.5.2021 г. 20:56	Папка с файлове	
img	13.3.2021 г. 7:00	Папка с файлове	
patient	15.5.2021 г. 16:44	Папка с файлове	
TCPDF	13.3.2021 г. 7:00	Папка с файлове	
contactFunc	15.5.2021 г. 13:13	PHP файл	1 КБ
contacts	10.5.2021 г. 20:40	PHP файл	2 КБ
footer	25.4.2021 г. 15:31	PHP файл	3 КБ
func2	24.5.2021 г. 16:32	PHP файл	2 КБ
header	10.5.2021 г. 19:44	PHP файл	4 КБ
index	10.5.2021 г. 19:45	PHP файл	20 КБ
logout1	13.3.2021 г. 7:00	PHP файл	2 КБ
README	24.5.2021 г. 20:41	MD файл	13 КБ
services	25.4.2021 г. 15:46	PHP файл	3 КБ
Signup	10.5.2021 г. 19:49	PHP файл	6 КБ

Снимка №2 - Файлова структура

1.2 Разработване

С цел поетапното запознаване с изграждането на системата са представени части от кода и ключовите функционалности на системата. Допълнително са включени някои от основните екрани, визуализиращи се пред различните типове потребители.

Дизайнът е изграден чрез HTML и CSS. Благодарение на използването на програмната рамка Bootstrap, системата се характеризира и с адаптивен дизайн. Разполагаме с няколко CSS файла, които са вградени в отделните страници. Използвани са два подхода на слагане на CSS – както външен, така и вграден подход.

Избраната технология за реализацията на логиката на уеб базираната система е PHP. Инсталиран е локален сървър WAMP, за да достъпваме проекта. Освен локалният сървър, е необходимо да имаме и среда за разработка (препоръчително VS Code или SublimeText), както и браузър.

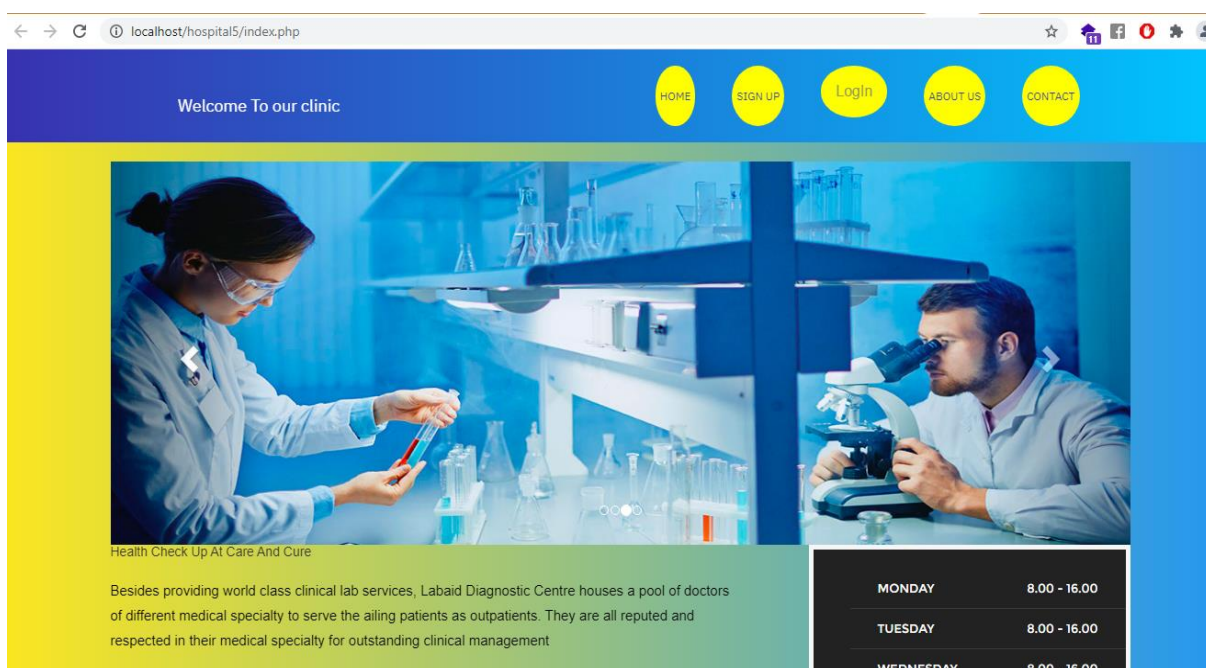
Застъпването на проекта става чрез изписването на localhost/името на папката на проекта в url-а на браузъра. Съответно при стартиране, сървърът ни препраща към index.php страницата, която се явява и начална за проекта.

2. Слой потребителски интерфейс

Презентационният слой включва презентационните елементи, които в случая на уеб приложение се създават с помощта на HTML, CSS и JavaScript. Този код е капсулиран в отделни файлове според тяхното предназначение. В прототипа е използван PHP като език за интерпретация на данните и изграждане на потребителския интерфейс. В отделни файлове се съхраняват HTML с PHP, CSS и JavaScript.

2.1 Начална страница

На началната страница на уеб системата разполагаме с навигационно меню, слайдър и няколко секции със съдържание. Част от началната страница може да бъде видяна на снимка №3.



Снимка №3 – Начална страница

Съдържащите се компоненти на началната страница са:

- Nav меню във header частта.

Всеки уебсайт е важно да има лесно за използване навигационно меню. Навигационната лента е изградена от HTML елементи. По конкретно, за направата на лентата за навигация е използван стандартен HTML списък. Навигационна лента е просто списък от линкове, затова е предпочетено използването на `` и `` тагове за оформление. Имаме и наложени CSS свойства за по-анимиран и добре изглеждащ вид.

- Слайдър, изграден посредством HTML, CSS и интегриран Bootstrap.

За да направим така, че слайдера да се активира автоматично след зареждане на страницата ни, добавяме HTML атрибута `data-ride` със стойност `carousel`. Извадка от кода за слайдера може да бъде видяна на снимка №4.

```
<div id="myCarousel" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#myCarousel" data-slide-to="0" class=""></li>
    <li data-target="#myCarousel" data-slide-to="1" class=""></li>
    <li data-target="#myCarousel" data-slide-to="2" class="active"></li>
    <li data-target="#myCarousel" data-slide-to="3" class=""></li>
  </ol>

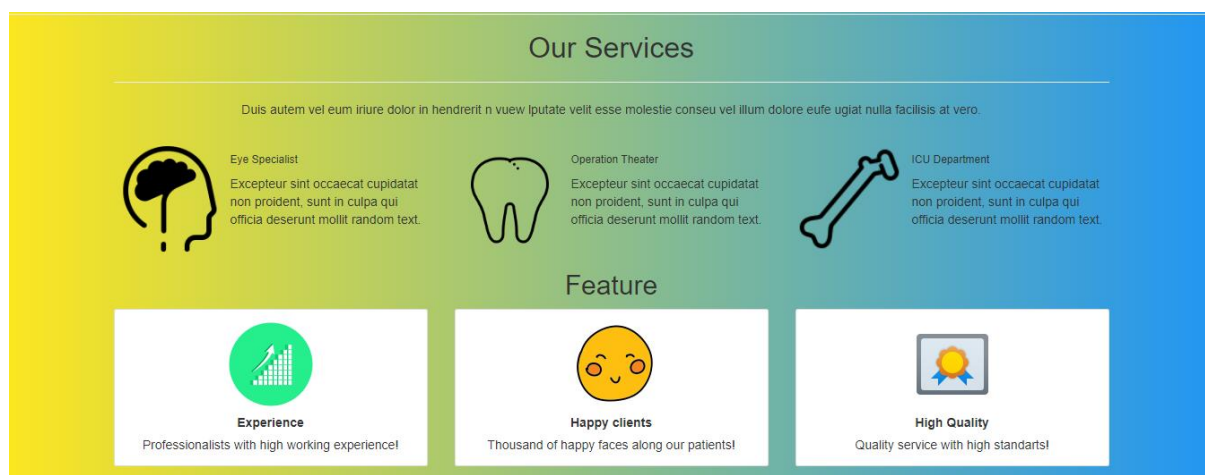
  <!-- Wrapper for slides -->
  <div class="carousel-inner">
    <div class="item active">
      
      <div class="carousel-caption">
        </div>
    </div>
  </div>

  <!-- Left and right controls -->
  <a class="left carousel-control" href="#myCarousel" data-slide="prev">
    <span class="glyphicon glyphicon-chevron-left"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="right carousel-control" href="#myCarousel" data-slide="next">
    <span class="glyphicon glyphicon-chevron-right"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Снимка №4 – Кодово представяне на слайдър

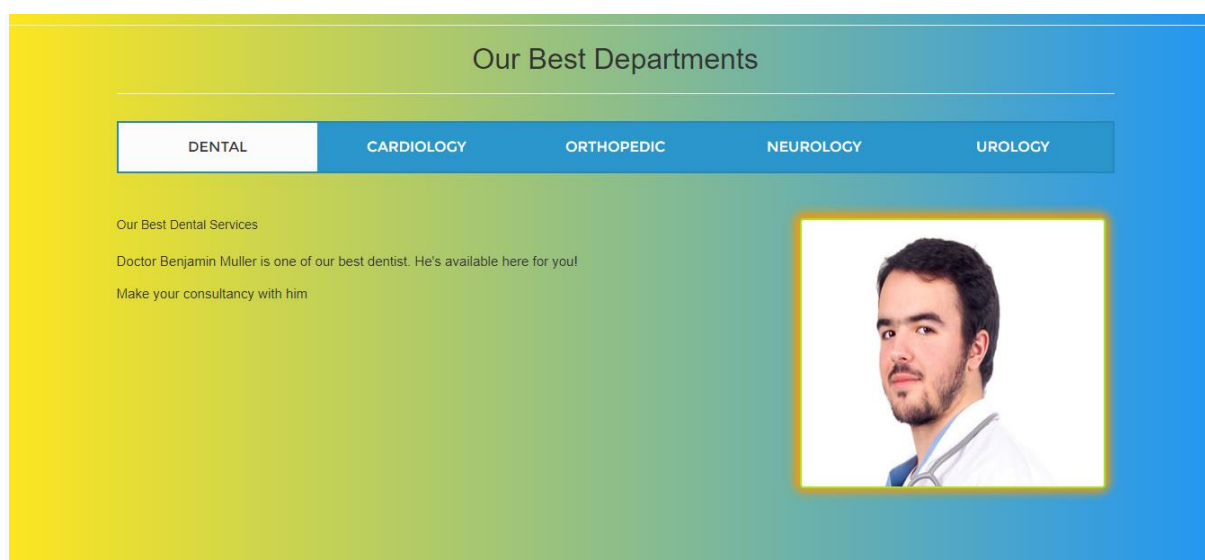
- Няколко секции с различно съдържание

Секцията “Our Services” представя някои от предлаганите от болничното заведение услуги и изтъква три характеристики на системата, а именно: професионализъм с голям опит, щастливи клиенти и високо качество. Изглед на тази секция може да открием на снимка №5.



Снимка №5 – Секция от началната страница

Следващата секция от началната страница е “Our Best Departments”. В нея са представени лекари от няколко различни специалности, например дентална медицина, кардиолози и т.н. Всеки лекар разполага със снимка и кратко описание за себе си. Екран от тази секция е показан на снимка №6.



Снимка №6 – Втора секция от начална страница

Тагът <section> дефинира секции в един HTML документ. Информацията в отделните секции е представена и с помощта на div тагове. <div> е блоков елемент, който може да се използва като контейнер, за групиране на други елементи. Той няма никакво специално значение или предназначение. Единствено, заради това, че е блок – браузъра ще направи нов ред преди и след него. Използван заедно със CSS, <div> елемента може да се използва за стилизиране на големи блокове от съдържание.

На началната страница разполагаме още с таблица с работните часове на лекарите и часовник, показващ часа в реално време, изграден посредством JavaScript. Кодът, използван за реализирането на тези два компонента от системата, е представен на снимка №7.

```
<div class="col-md-4">
  <div class="timing"> <i class="lnr lnr-clock"></i>
    <ul>
      <li> Monday <span>8.00 - 16.00</span></li>
      <li> Tuesday <span>8.00 - 16.00</span></li>
      <li> Wednesday <span>8.00 - 16.00</span></li>
      <li> Thursday <span>8.00 - 16.00</span></li>
      <li> Friday <span>8.00 - 16.00</span></li>
      <li> Saturday <span>8.00 - 16.00</span></li>
      <li> Sunday <span>8.00 - 16.00</span></li>
    </ul>
  </div>
  <h1 class="clocktxt">The time is</h1>
  <div id="clock">--- PM</div>
```

```
function clock(){

//Save the times in variables

let today = new Date();

let hours = today.getHours();
let minutes = today.getMinutes();
let seconds = today.getSeconds();

//Set the AM or PM time

if (hours >= 12){
meridiem = " PM";
}
else {
meridiem = " AM";
}
```

```

//convert hours to 12 hour format and put 0 in front
if (hours>12){
hours = hours - 12;
}
else if (hours===0){
hours = 12;
}

//Put 0 in front of single digit minutes and seconds

if (minutes<10){ minutes="0" + minutes; } else { minutes=minutes; } if (seconds<10){ seconds="0" + seconds; } else { seconds=seconds; } document.getElementById("clock").innerHTML=(hours + ":" + minutes + ":" + seconds + meridiem); }
setInterval('clock()', 1000);

```

Снимка №7 – Таблица с работни часове и часовник

Друга важна част от системата са частта header и частта footer. Извикваме header.php и footer.php съответно най-отгоре в документа и най-отдолу посредством служебна дума include.

В header.php, и по-конкретно в тага <head> интегрираме нашият CSS файл, а също така и Bootstrap, както може да видим на снимки №8 и №9.

```
<link rel="stylesheet" href="assets/css/stylein.css">
```

Снимка №8 – Интеграция на css файл

```

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-
BVYiSiFeK1dGmJRAkycuHAnRg320mUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin=
"anonymous">

```

Снимка №9 – Интеграция на Bootstrap

В footer.php са изградени две колони, една за контакти и друга за авторски права.

Например колоната за контакти изглежда по следния начин:

```

<div class="col-lg-3 col-md-6 footer-info">
  <h3>Contacts</h3>
  <p>
    Plv, ulitsa.... <br>

    <strong>Phone:</strong> <a href="tel:+60133092691">+359 22 328329</a>
  <br>
    <strong>Email:</strong> <a href="mailto:info@tixedu.com">ClinicPlv@gmail.com</a> <br>

```

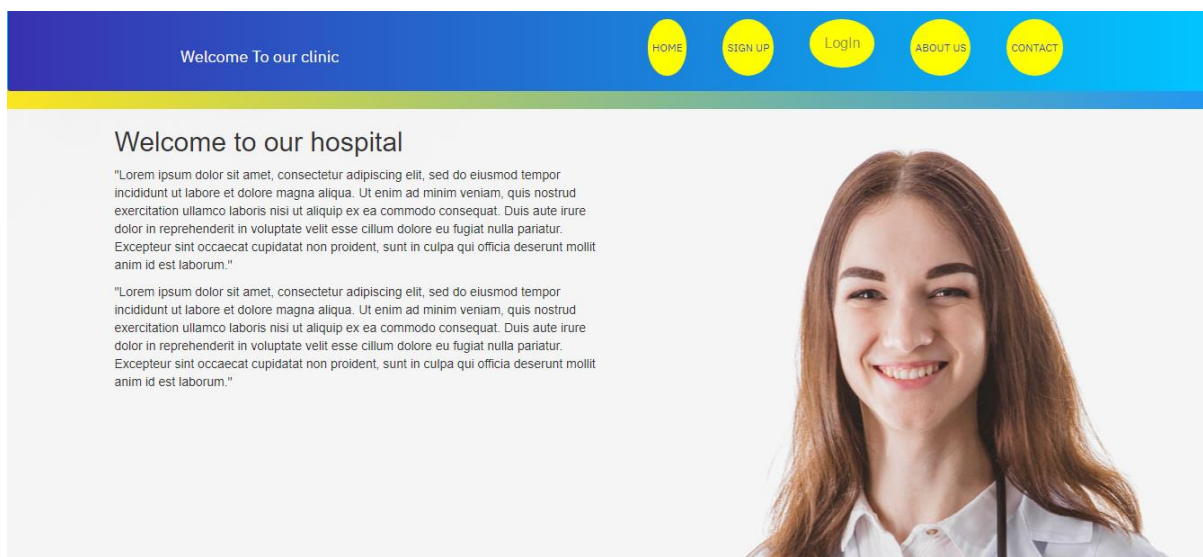
```
</p>  
</div>
```

Снимка №10 – Колона за контакти

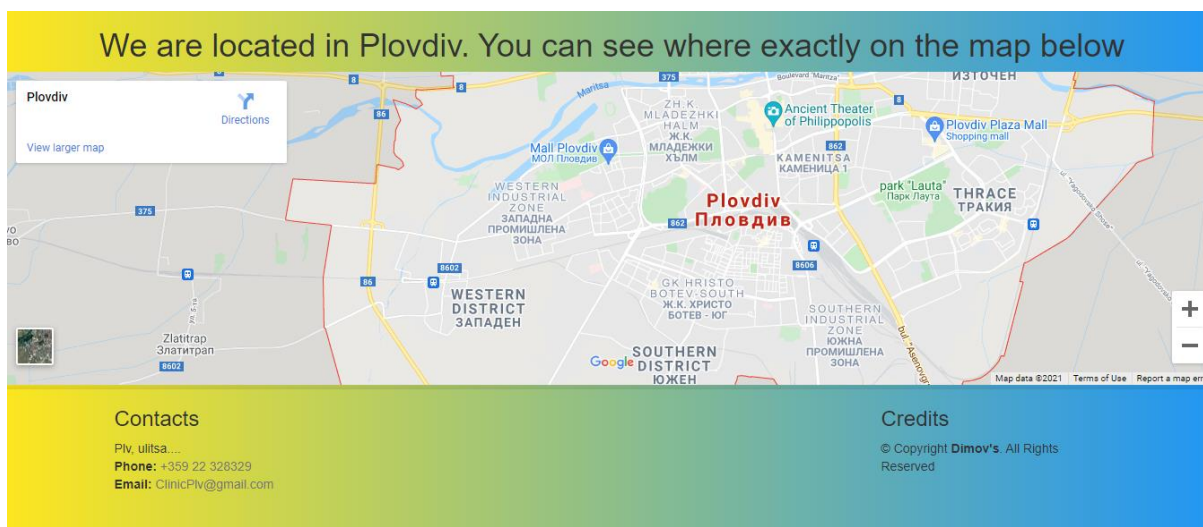
Началната страница, както и намиращите се в Nav менюто страници, About Us и Contact, са единствените страници в уеб системата, които могат да бъдат достъпни и от така наречения non-authenticated user (нерегистриран потребител). За достъп до всички останали страници, е нужно потребителят да влезе в акаунта си през страницата за вход. Също така, има и връзка в менюто, която води до страницата за регистрация, в случай, че потребителят все още няма профил в системата.

2.2 Страница “За нас”

В страница “About us” разполагаме с описание относно болницата и Google Map локация на картата. Изглед на страницата може да бъде намерен на снимки №11 и №12.



Снимка №11 – Страница “За нас”

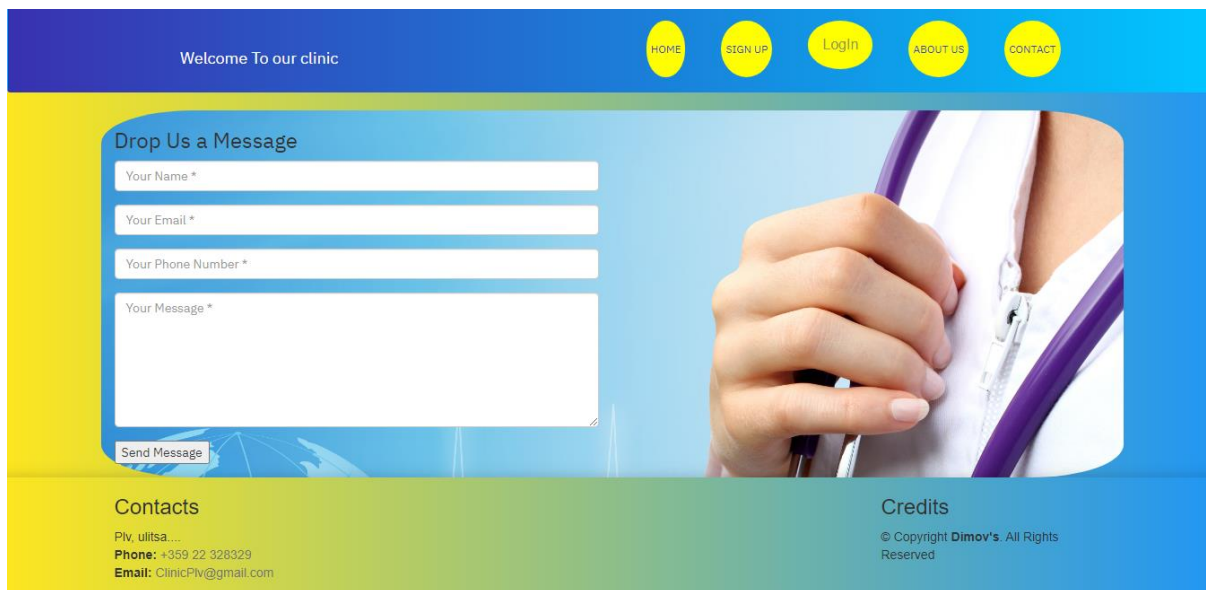


Снимка №12 – Локация

Интегрирането на карта става посредством използването на така наречения iframe HTML таг, който използва източник за информационен поток. Като източник (src) използваме Google maps генерираната локация, поставяйки я вътре в тага iframe.

2.3 Страница “Контакти”

Тази страница съдържа контактна форма, чрез която нерегистриран потребител може да се свърже с администрацията на системата. На снимка №13 можем да видим и снимка от екрана на тази страница.



Снимка №13 – Страница “Контакти”

Изпращаме данните попълнени във формата посредством следния php код:

Използваме include, за да свържем файла към базата данни:

```
<?php
include "db/config.php";

if (isset($_POST['btnSubmit'])) {
    $name = $_POST['txtName'];
    $email = $_POST['txtEmail'];
    $contact = $_POST['txtPhone'];
    $message = $_POST['txtMsg'];

    $query = "insert into contact(name,email,contact,message) values('$name', '$email', '$contact', '$message')";
    $result = mysqli_query($con, $query);

    if ($result) {
        echo '<script type="text/javascript">';
```

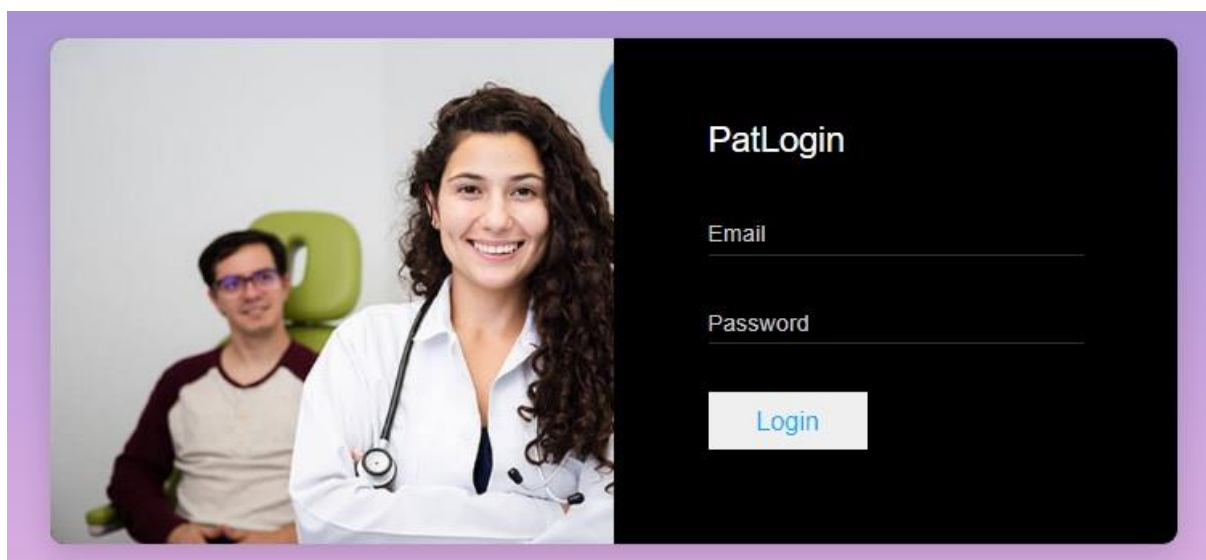
```
        echo 'alert("Message sent successfully!");';  
        echo 'window.location.href = "contacts.php"';  
        echo '</script>';  
    }  
}
```

Снимка №14 – Формуляр за изпращане на писмо до администрация

2.4 Ауторизация

2.4.1 Вход

Страницата за вход е част от системата, която обработва упълномощаването на потребителите да приемат конкретни привилегии (роли) след влизане в акаунт. Налични са три потребителски роли в системата, които са съответно пациент, доктор и администратор. Изглед на страницата е показан на снимки №15 и №16.

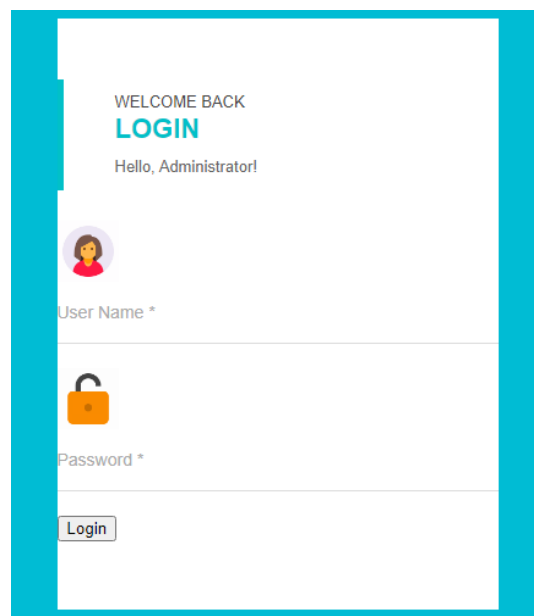


Снимка №15 – Страница за вход в системата

За да може потребителят на системата да попълни своите идентификационни данни, са подготвени две полета за въвеждане, които да приемат потребителско име (за пациенти - имейл) и парола. Разполагаме с три различни интерфейса за всяка една отделна категория потребители.

След като бъде натиснат бутон “Login”, ще бъде извършено упълномощаване въз основа на предоставените идентификационни данни, чрез комуникация между системата и базата данни, обработваща потребителски акаунти. В случай, че не бъде намерен клиентски акаунт на база предоставените идентификационни данни, ще се появи предупредително съобщение, което ще съобщи на потребителя за невалидни данни (било то потребителско име или парола), като по този начин ще бъде забранено на потребителя да влезе в системата.

Клиентът ще трябва и да опита отново с коригирани идентификационни данни. За съжаление, засега системата не разполага с процедура за възстановяване на парола и ако е необходимо възстановяване, клиентът трябва да се свърже лично с администратор, за да може вторият да предостави нова парола на потребителя. Все пак, ако идентификационните данни съвпадат с точен запис в базата данни, тогава упълномощаването ще бъде завършено успешно и потребителят ще получи достъп до уебсайта. След успешно упълномощаване клиентът ще бъде пренасочен към своето табло (Dashboard), като клиентът има текуща сесия към системата.



Снимка №16 – Вход за админ

Ако разгледаме и част от кода използван за реализация на формата на снимка №17:

```
<form class="form" role="form" method="post" action="func.php" accept-
charset="UTF-8">

    <div class="input-group">
        <input class="input--style-
3" type="text" placeholder="Email" name="email" onkeydown="return alphaOnly(ev
ent);" required />

    </div>
    <div class="input-group">
        <input class="input--style-
3" type="password" placeholder="Password" name="password2" required />
    </div>
    <div class="p-t-10">
        <button class="btn btn-info btn-
block login" type="submit" name="patsub" value="Login">Login</button>
    </div>
</form>
```

Снимка №17 – Формуляр за вход в системата

Също и функционалността:

```
<?php
session_start();
$con = mysqli_connect("localhost", "root", "", "myhmsdb");
if (isset($_POST['patsub'])) {
    $email = $_POST['email'];
    $password = $_POST['password2'];
```

```

$query = "select * from patreg where email='$email' and password='$password'";
$result = mysqli_query($con, $query);
if (mysqli_num_rows($result) == 1) {
    while ($row = mysqli_fetch_array($result, MYSQLI_ASSOC)) {
        $_SESSION['pid'] = $row['pid'];
        $_SESSION['username'] = $row['fname'] . " " . $row['lname'];
        $_SESSION['fname'] = $row['fname'];
        $_SESSION['lname'] = $row['lname'];
        $_SESSION['gender'] = $row['gender'];
        $_SESSION['contact'] = $row['contact'];
        $_SESSION['email'] = $row['email'];
    }
    header("Location:patient-panel.php");
} else {
    echo ("<script>alert('Invalid Username or Password. Try Again!');
        window.location.href = 'patientlogin.php';</script>");
}
}

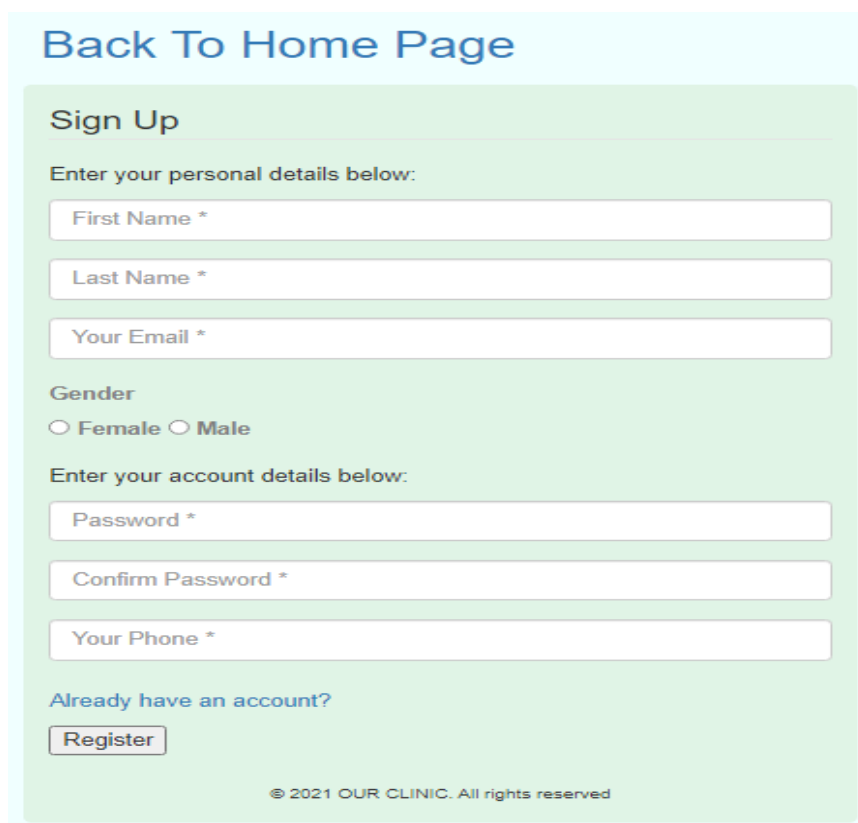
```

Снимка №18 – PHP код за реализиране на вход

Забелязваме, че имаме стандартна форма за попълване на имейл и парола, която се изпраща до сървъра. При наличие на получена POST заявка, на две променливи присвояваме получените стойности и ги изпращаме към сървъра. Ако върнатата стойност е истина, значи че въведените от потребителя данни са регистрирани в системата. В променливата `$_SESSION` записваме някой от данните за влезналия потребител. Тези данни помагат за “предвиждането” на потребителя през системата.

2.4.2 Регистрация

Регистрационната форма е само за пациенти, които искат да направят акаунт в системата. В страницата за регистрация потребителят на системата трябва да може да извърши вмъкване на информация в полета, които представляват име, фамилия, имейл, пол, телефонен номер, парола и повторна парола като потвърждение. На снимка №19 е изобразена и страницата за регистрация.



Снимка №19 – Страница за регистрация

Клиентът може да регистрира акаунт в системата, като щракне върху бутонът за регистрация в долната част на формата. След това регистрационният формуляр изпраща съобщение за успешно завършена регистрация, ако данните са въведени коректно. При некоректно въведени данни, системата ще изведе съобщение за грешка. Например, когато бъде прихванато несъответствие на паролите, на екрана ще се появи предупреждение за уведомяване на клиента за несъответстващи пароли. Също така, разполагаме с регулярни изрази за паролата, с правила, като например да е повече от 6 символа и други. В случай, че потребителят е забравил да попълни някое от задължителните полета след натискане на бутона „Регистрация“, ще се появи изскачащ прозорец върху съответното непопълнено поле. Ако клиентът извърши вмъкването на данни, без да задейства грешки, той ще бъде уведомен за успешна регистрация, след като бъде натиснат бутонът „Регистрация“. Акаунтът също така успешно бива добавен

към базата данни. Кодът, използван за реализиране на формуляра за регистрация е представен на снимка №20.

```
<form method="post" action="func2.php">

  <fieldset>
    <legend>
      Sign Up
    </legend>
    <p>
      Enter your personal details below:
    </p>
    <div class="form-group">
      <input type="text" class="form-
control" placeholder="First Name *" name="fname" onkeydown="return alphaOnly(e
vent);" required />
    </div>
    <div class="form-group">
      <input type="text" class="form-
control" placeholder="Last Name *" name="lname" onkeydown="return alphaOnly(ev
ent);" required />
    </div>
    <div class="form-group">
      <input type="email" class="form-
control" placeholder="Your Email *" name="email" />
    </div>
    <div class="form-group">
      <label class="block">
        Gender
      </label>
      <div class="clip-radio radio-primary">
        <input type="radio" id="rg-female" name="gender" value="Female">
        <label for="rg-female">
          Female
        </label>
        <input type="radio" id="rg-male" name="gender" value="Male">
        <label for="rg-male">
          Male
        </label>
      </div>
    </div>
    <p>
      Enter your account details below:
    </p>
    <div class="form-group">
      <span class="input-icon">
        <input type="password" class="form-
control" placeholder="Password *" id="password" name="password" onkeyup='check
();' required />
      </span>
    </div>
  </fieldset>
</form>
```

```

</div>
<div class="form-group">
    <span class="input-icon">
        <input type="password" class="form-
control" id="cpassword" placeholder="Confirm Password *" name="cpassword" onke
yup='check();' required /><span id='message'></span>
    </div>

    <div class="form-group">
        <input type="tel" minlength="10" maxlength="10" name="contact" class="fo
rm-control" placeholder="Your Phone *" />
    </div>
<div class="form-actions">
    <p>
        <a href="patient/patientlogin.php">Already have an account?</a>
    </p>
    <input type="submit" class="btnRegister" name="patsub1" onclick="return
checklen();" value="Register" />
</div>
</fieldset>
</form>

```

Снимка №20 – Формуляр за регистрация

Файлът FuncReg.php съдържа следния кодов вид:

```

<?php
session_start();
$con=mysqli_connect("localhost","root","","myhmsdb");
if(isset($_POST['patsub1'])){
    $fname=$_POST['fname'];
    $lname=$_POST['lname'];
    $gender=$_POST['gender'];
    $email=$_POST['email'];
    $contact=$_POST['contact'];
    $password=$_POST['password'];
    $cpassword=$_POST['cpassword'];
    if($password==$cpassword){
        $query="insert into patreg(fname,lname,gender,email,contact,password,cpass
word) values ('$fname','$lname','$gender','$email','$contact','$password','$cp
assword')";
        $result=mysqli_query($con,$query);
        if($result){
            $_SESSION['username'] = $_POST['fname']." ".$_POST['lname'];
            $_SESSION['fname'] = $_POST['fname'];
            $_SESSION['lname'] = $_POST['lname'];
            $_SESSION['gender'] = $_POST['gender'];
            $_SESSION['contact'] = $_POST['contact'];

```

```
$_SESSION['email'] = $_POST['email'];  
header("Location:patient/patient-panel.php");  
}
```

Снимка №21 – PHP код за реализиране на регистрацията

Дефинирана е форма, която позволява изпращането на данни към web сървъра. В случая имаме данни, касаещи регистрацията на потребител в системата. Методът на изпращане е POST, при който данните към сървъра се изпращат самостоятелно. Елементът <input> позволява въвеждането на информация от страна на потребителя. Атрибута type задава типа на елемента, който се въвежда, а name задава името на елемента като име на променлива към сървъра. Бутоните се асоциират с определени действия при кликане. Бутонът ratsub1 изпраща въведените във входните полета данни към сървъра.

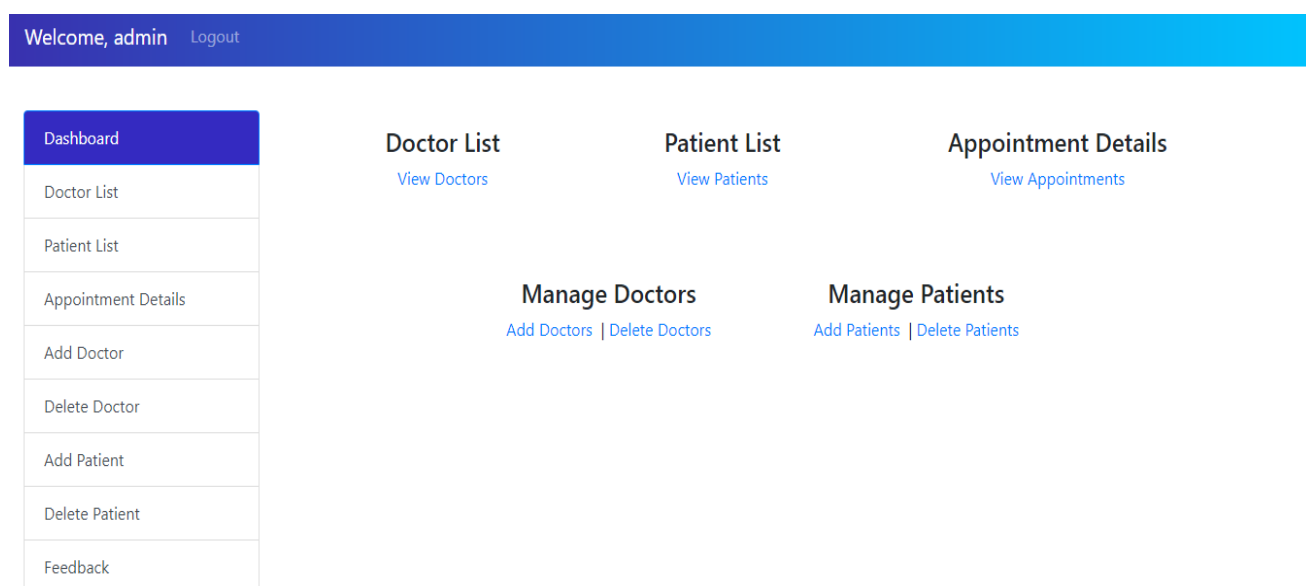
Ако имаме POST заявка функцията ще обработи входните данни. На променливите присвояваме стойността от получената POST заявка със съответните атрибути. Данните, които получаваме от формата трябва да бъдат точно такива, каквито очакваме. В случая, този критерий е сведен до определяне на дължината на получените данни. Например, проверяваме дали броят на символите за паролата е по-голям от 3 символа. Ако това условие е изпълнено, няма да имаме ValidationError и продължаваме нататък. Извеждаме съобщение за успешна регистрация и препращаме потребителя към собственото си табло (dashboard).

2.5 Потребителски групи

Както стана ясно, има три категории за потребители – Администратор, доктор и пациент. Функциите на съответните категории ще варират в зависимост от потребителския тип.

2.5.1 Администратор

Администраторът (или Ръководството) разполага с най-много функции в системата. След ауторизация в системата, администраторът получава достъп до своето табло. Изглед на таблото е показан на снимка №22.



Снимка №22 – Екран на табло за администратор

Администраторът има привилегиите да създава потребителски профили както за пациенти, така и за лекари. Извадка от двата екраните може да бъде видяна на снимки №23 и №24.

This screenshot shows the 'Add Doctor' form in the admin dashboard. At the top, a blue header bar contains the text 'Welcome, admin' and a 'Logout' link. On the left, a vertical sidebar menu lists various dashboard options: 'Dashboard', 'Doctor List', 'Patient List', 'Appointment Details', 'Add Doctor' (which is highlighted in blue), 'Delete Doctor', 'Add Patient', 'Delete Patient', and 'Feedback'. The main content area contains a form for adding a new doctor. The form fields are: 'Doctor Name:' (text input), 'Specialization:' (dropdown menu with 'Select Specialization' and a downward arrow), 'Email ID:' (text input), 'Password:' (text input), and 'Confirm Password:' (text input). A blue 'Add Doctor' button is positioned below the form fields.

Снимка №23 – Екран за добяване на лекар

This screenshot shows the 'Add Patient' form in the admin dashboard. It features the same blue header bar with 'Welcome, admin' and 'Logout'. The sidebar menu on the left is identical to the previous screenshot, but 'Add Patient' is now highlighted in blue. The main form area includes fields for: 'First Name:' (text input), 'Last Name:' (text input), 'Gender:' (dropdown menu with 'Select gender' and a downward arrow), 'Email ID:' (text input), 'Phone:' (text input), 'Password:' (text input), and 'Confirm Password:' (text input). A blue 'Add Patient' button is located at the bottom of the form.

Снимка №24 – Екран за добяване на пациент

Кодът, използван за направата на този формуляр е следният:

```
<form class="form-group" method="post" action="admin-panel.php">
  <div class="row">
    <div class="col-md-4"><label>Doctor Name:</label></div>
    <div class="col-md-8"><input type="text" class="form-
control" name="doctor" onkeydown="return alphaOnly(event);" required></div><br>
    <div class="col-md-4"><label>Specialization:</label></div>
    <div class="col-md-8">
      <select name="special" class="form-
control" id="special" required="required">
        <option value="head" name="spec" disabled selected>Select Specializati
on</option>
        <option value="General" name="spec">General</option>
        <option value="Cardiologist" name="spec">Cardiologist</option>
        <option value="Neurologist" name="spec">Neurologist</option>
        <option value="Pediatrician" name="spec">Pediatrician</option>
        <option value="testcian" name="spec">Test</option>
      </select>
    </div><br><br>
    <div class="col-md-4"><label>Email ID:</label></div>
    <div class="col-md-8"><input type="email" class="form-
control" name="demail" required></div><br><br>
    <div class="col-md-4"><label>Password:</label></div>
    <div class="col-md-8"><input type="password" class="form-
control" onkeyup='check();' name="dpassword" id="dpassword" required></div><br>
    <div class="col-md-4"><label>Confirm Password:</label></div>
    <div class="col-md-8" id='cpass'><input type="password" class="form-
control" onkeyup='check();' name="cdpassword" id="cdpassword" required>&nbsp; &
nbsp<span id='message'></span> </div><br><br>

    <br><br>
  </div>
  <input type="submit" name="docsub" value="Add Doctor" class="btn btn-
primary">
</form>
```

Снимка №25 – Формуляр за добавяне на лекар

Изградена е форма за добавяне на лекар посредством HTML, използваща метод POST. Разполагаме с полета за въвеждане на данни за доктор. Необходимо е да бъдат попълнени всички изисквани данни, за да може да записът да бъде добавен успешно към базата данни.

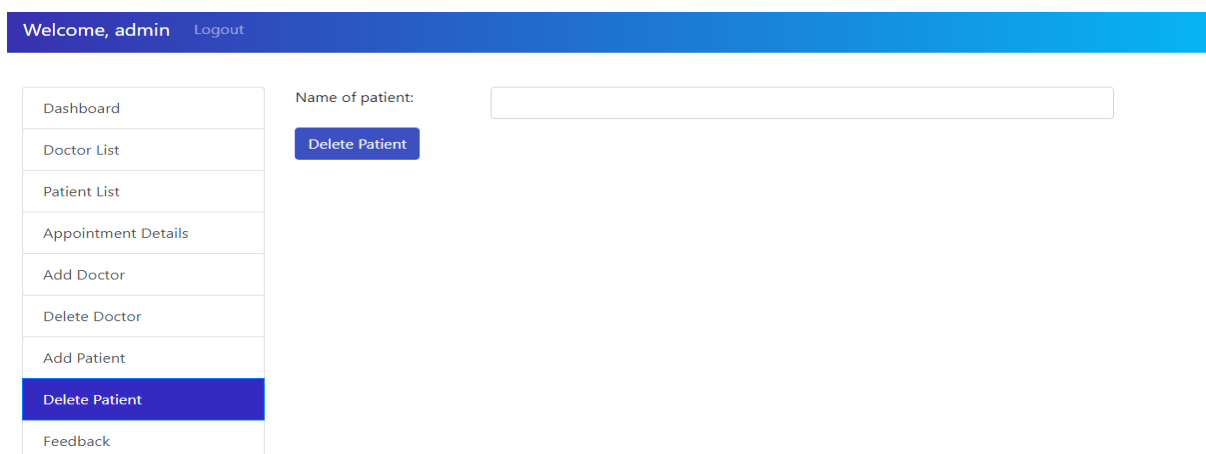
Използвана е и следната PHP валидация:

```
<?php
if(isset($_POST['docsub']))
{
$doctor=$_POST['doctor'];
$password=$_POST['password'];
$email=$_POST['email'];
$spec=$_POST['special'];
$query="insert into doctb(username,password,email,spec)values('$doctor','$password','$email','$spec')";
$result=mysqli_query($con,$query);
if($result)
{
echo "<script>
    alert('Doctor added successfully!');
</script>";
}
}
```

Снимка №26 – PHP код за реализиране на добавяне на лекар

При натискане на бутона данните се изпращат към базата данни и записът се записва.

Също така администраторът има опции да изтрива пациенти и лекари. Екран за тази функционалност е показан на снимка №27.



Снимка №27 – Екран за изтриване на пациент

Кодът, който използва формуляра за изтриване на доктор е следния:

```
if(isset($_POST['docsub1']))
{
$doctor=$_POST['doctor'];
$query="delete from doctb where username='$doctor'";
$result=mysqli_query($con,$query);
if($result)
{
echo "<script>
    alert('Doctor removed successfully!');
</script>";
}
else{
echo "<script>
    alert('Unable to delete!');
</script>";
}
}
```

Снимка №28 – Формуляр за изтриване на пациент

Проверяваме дали бутона docsuble натиснат и извличаме името на доктора от таблицата doctb в базата. Използваме условен оператор if else, при който ако резултатът е успешен излиза съобщение за прогрес, в противен случай се появява съобщение за грешка.

Администраторът разполага и с пълен списък от съществуващи пациенти и доктори в системата.

Също така админът може да използва търсачка (по имейл за лекари и по телефонен номер за пациенти), докато е в менюто със списъците. Визуализация на екран за търсене е представена на снимка №29.

Username	spec	Email
Ivanov	General	ivanov@gmail.com
Back to dashboard		

Снимка №29 – Екран за търсене на лекар

Изграден е и следният формуляр за търсенето по критерий:

```
if(isset($_POST['doctor_search_submit']))
{
    $contact=$_POST['doctor_contact'];
    $query = "select * from doctb where email= '$contact'";
    $result = mysqli_query($con,$query);
    $row=mysqli_fetch_array($result);
    if($row['username']=="" & $row['spec']=="" & $row['email']==""){
    echo "<script>
        alert('No entries found!');
        window.location.href = 'admin-panel.php#list-doc';
    </script>";
    }
    else {
    echo "<div class='container-fluid' style='margin-top:50px;'>
        <div class='card'>
            <div class='card-body' style='background-color:#342ac1;color:#ffffff;'>
                <table class='table table-hover'>
                    <thead>
                        <tr>
                            <th scope='col'>Username</th>
                            <th scope='col'>spec</th>
                            <th scope='col'>Email</th>
                        </tr>
                    </thead>
                    <tbody>";
                </table>
                <center><a href='admin-panel.php' class='btn btn-
light'>Back to dashboard</a>
            </div>
            </center>
        </div>
    </div>
    </div>";
    }
}
?>
```

Снимка №30 – Формуляр за търсене на лекар

Извличаме въведения имейл адрес от таблицата doctb и проверяваме дали съществуват налични записи. За целта използваме условен оператор if, ако съществуват записи, те се визуализират в таблица, в противен случай излиза съобщение за грешка. По аналогичен начин става и търсенето за пациенти.

Друга опция на администраторът е да преглежда историята на записаните часове. Там той разполага с цялостна информация за записания час, например кой е пациентът, кой е неговият обслужващ лекар и т.н. Формуляр за тази функционалност е показан на снимка №31.

```
<table class="table table-hover">
  <thead>
    <tr>
      <th scope="col">Appointment ID</th>
      <th scope="col">Patient ID</th>
      <th scope="col">First Name</th>
      <th scope="col">Last Name</th>
      <th scope="col">Gender</th>
      <th scope="col">Email</th>
      <th scope="col">Contact</th>
      <th scope="col">Doctor Name</th>
      <th scope="col">Appointment Date</th>
      <th scope="col">Appointment Time</th>
      <th scope="col">Appointment Status</th>
    </tr>
  </thead>
  <tbody>
    <?php

    $con = mysqli_connect("localhost", "root", "", "myhmsdb");
    global $con;

    $query = "select * from appointmenttb;";
    $result = mysqli_query($con, $query);
    while ($row = mysqli_fetch_array($result)) {
      ?>
      <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['pid']; ?></td>
        <td><?php echo $row['fname']; ?></td>
        <td><?php echo $row['lname']; ?></td>
        <td><?php echo $row['gender']; ?></td>
        <td><?php echo $row['email']; ?></td>
        <td><?php echo $row['contact']; ?></td>
        <td><?php echo $row['doctor']; ?></td>
        <td><?php echo $row['appdate']; ?></td>
        <td><?php echo $row['apptime']; ?></td>
        <td>
          <?php if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 1))
{
          echo "Active";
        }
      </td>
    </tr>
  </tbody>
</table>
```

```

        if (($row['userStatus'] == 0) && ($row['doctorStatus'] == 1)) {
            echo "Cancelled by Patient";
        }

        if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 0)) {
            echo "Cancelled by Doctor";
        }
        ?></td>
    </tr>
<?php } ?>
</tbody>
</table>

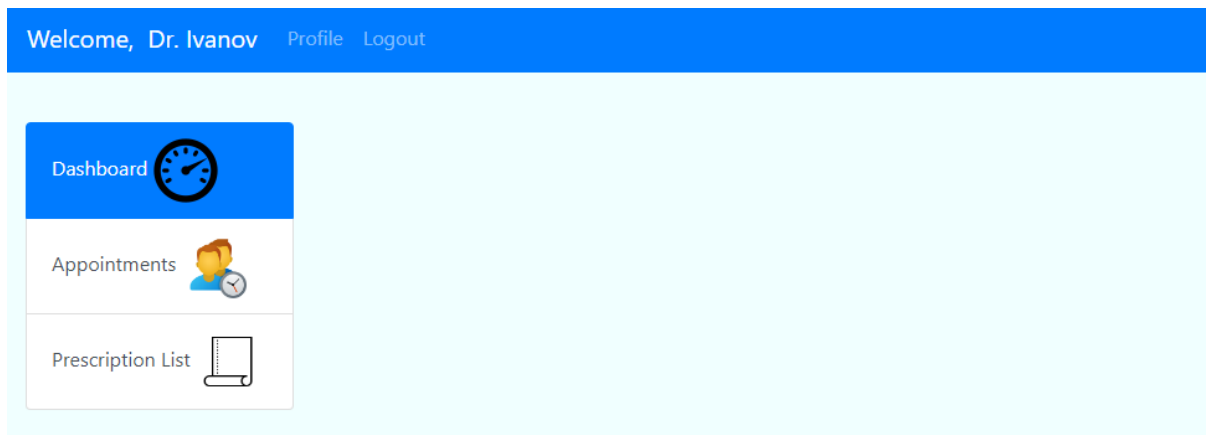
```

Снимка №31 – Формуляр със списък с минали прегледи

От менюто Feedback администраторът може да преглежда съобщения от нерегистрирани потребители в системата. Тук той също разполага с търсачка, ако желае по-бързо да намери даден запис.

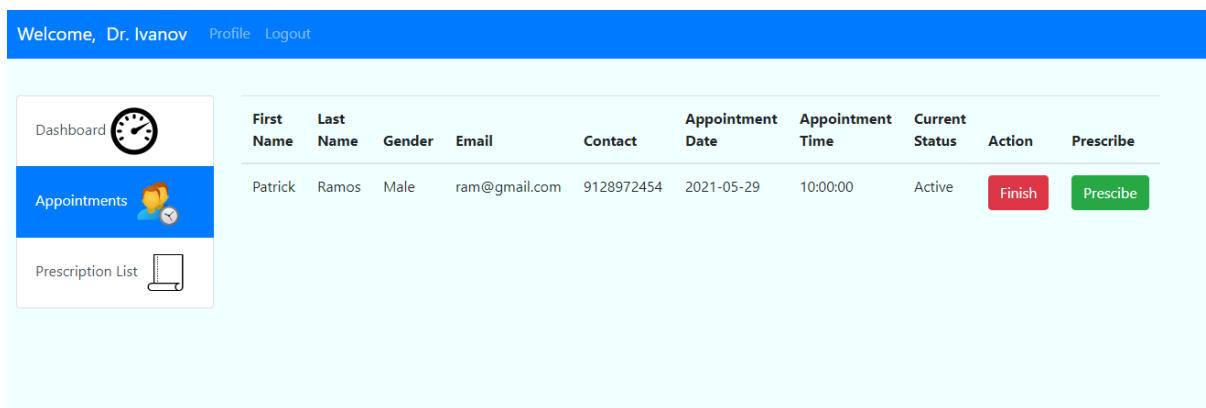
2.5.2 Доктор

Докторите разполагат с няколко менюта. Всеки доктор принадлежи към дадена специалност (Например уролог, дентална медицина, общо практикуващ лекар и т.н). Екран на таблото за лекар е представено на снимка №32.



Снимка №32 – Екран на табло за лекар

Докторите имат опцията да проверяват за насрочени часове от пациенти. Визуализация на тази функционалност е изобразена на снимка №33.



Снимка №33 – Екран със списък с насрочени часове

При натискане на бутона Prescribe, ще се зареди нова форма с три полета за въвеждане, които са респективно вида на заболяване, алергии и предписание (под формата на бележка).

Global Hospital
Logout
Back

Disease:

Allergies:

Prescription:

Prescribe

Снимка №34 – Екран за предписване на бележки

След като докторът въведе данните в посочените три форми, той ще може да приключи насрочения преглед чрез натискането на бутона Finish. Това автоматично ще изпрати предписаните данни от доктора до съответния пациент. Данните също биват записани в базата данни при натискане на бутона.

Друга опция на докторите им позволява да преглеждат лист със вече отминали прегледи (снимка №35).

Dashboard

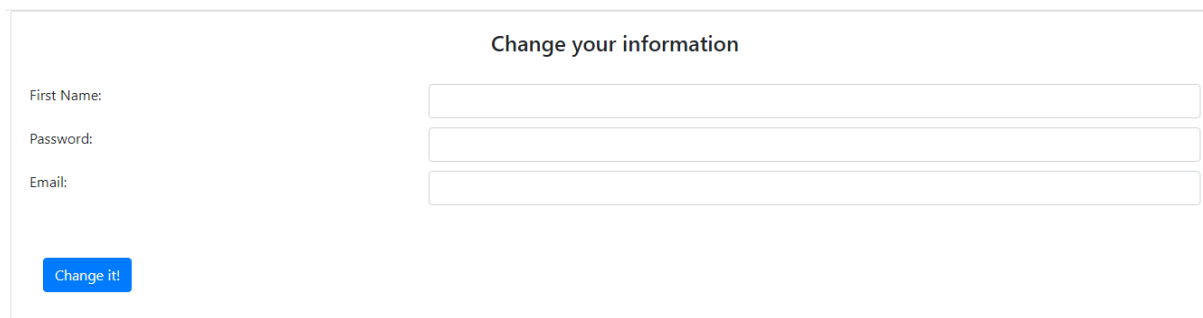
Appointments

Prescription List

First Name	Last Name	Appointment Date	Appointment Time	Disease	Allergy	Prescribe
Patrick	Ramos	2021-05-29	10:00:00	Cold	No	Antibiotic + a lot of watter.

Снимка №35 – Екран със списък на отминали прегледи

Докторите също така могат да коригират данни за потребителския си акаунт чрез бутона Profile, намиращ се в Nav менюто.



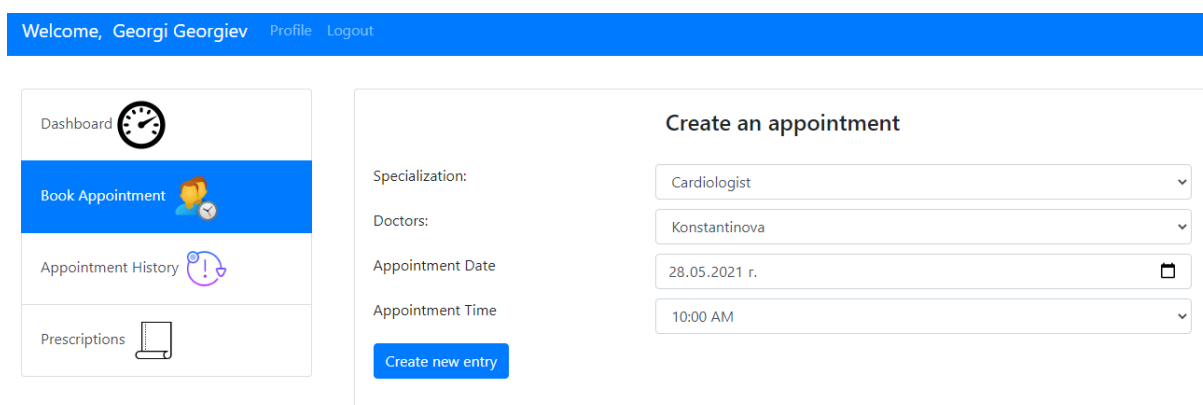
The screenshot shows a web form titled "Change your information". It contains three input fields for "First Name:", "Password:", and "Email:". Below these fields is a blue button labeled "Change it!".

Change your information	
First Name:	<input type="text"/>
Password:	<input type="password"/>
Email:	<input type="text"/>
<input type="button" value="Change it!"/>	

Снимка №36 – Екран за промяна на потребителски данни

2.5.3 Пациент

Пациентите разполагат с функции, базирани на техните нужди. Те могат да си насрочват часове за преглед. Необходимо е да изберат специалност, лекуващ лекар от съответната специалност, дата за преглед и час. Ако пациентът избере вече минала дата, ще бъде изведено съобщение за грешка. Също, ако за избраната дата е зает часът, избран от потребителя, отново ще излезе съобщение за грешка. При коректно въведени данни, часът ще бъде записан и изпратен до лекаря. Екран за записване на час за преглед е представен на снимка №37.



Снимка №37 – Екран за записване на час за преглед

Формулярът за горния изглед представлява:

```
<?php
if(isset($_POST['app-submit']))
{
    $pid = $_SESSION['pid'];
    $username = $_SESSION['username'];
    $email = $_SESSION['email'];
    $fname = $_SESSION['fname'];
    $lname = $_SESSION['lname'];
    $gender = $_SESSION['gender'];
    $contact = $_SESSION['contact'];
    $doctor=$_POST['doctor'];
    $email=$_SESSION['email'];

    $appdate=$_POST['appdate'];
    $apptime=$_POST['apptime'];
    $cur_date = date("Y-m-d");
    date_default_timezone_set('Europe/Sofia');
    $cur_time = date("H:i:s");
```

```

$apptime1 = strtotime($apptime);
$appdate1 = strtotime($appdate);

if(date("Y-m-d",$appdate1)>=$cur_date){
if((date("Y-m-
d",$appdate1)==$cur_date and date("H:i:s",$apptime1)>$cur_time) or date("Y-m-
d",$appdate1)>$cur_date) {
$check_query = mysqli_query($con,"select apptime from appointmenttb where doct
or='$doctor' and appdate='$appdate' and apptime='$apptime'");

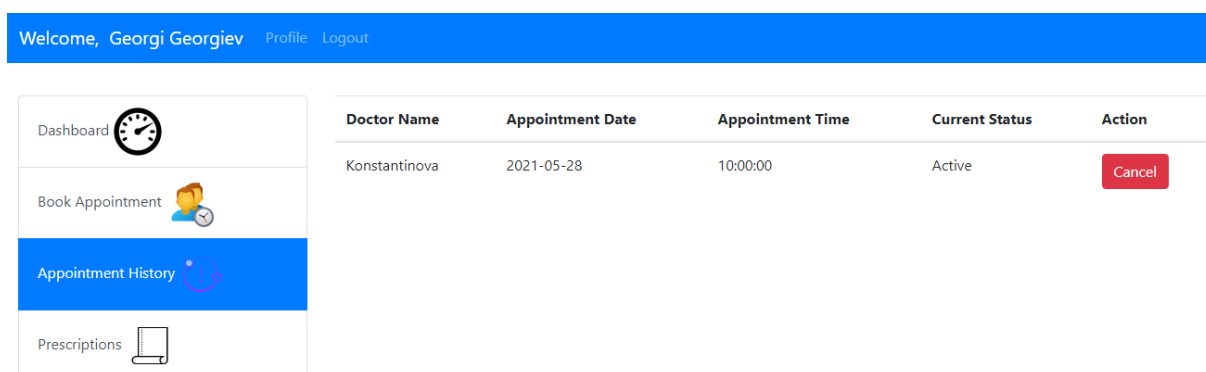
if(mysqli_num_rows($check_query)==0){
$query=mysqli_query($con,"insert into appointmenttb(pid,fname,lname,gender,ema
il,contact,doctor,appdate,apptime,userStatus,doctorStatus) values($pid,'$fname
','$lname','$gender','$email','$contact','$doctor','$appdate','$apptime','1','
1')");

if($query)
{
echo "<script>
    alert('Your appointment successfully booked');
</script>";
}
else{
echo "<script>
    alert('Unable to process your request. Please try again!');
</script>";
}
}
else{
echo "<script>
    alert('We are sorry to inform that the doctor is not available in this time
or date. Please choose different time or date!');
</script>";
}
}
else{
echo "<script>
    alert('Select a time or date in the future!');
</script>";
}
}
}

```

Снимка №38 – Формуляр за записване на час за преглед

През секцията Appointments History, пациентът ще може да види списък със своите прегледи. При неотложни ангажменти или спешни причини, пациентът може да откаже своя час чрез натискане на бутона Cancel. Тази функционалност е показана на снимка №39.



Снимка №39 – Екран със списък на прегледи

```
<?php
if (isset($_GET['cancel'])) {
    $query = mysqli_query($con, "update appointmenttb set userStatus='0' where ID = '" . $_GET['ID'] . "'");
    if ($query) {
        echo "<script>alert('Your appointment successfully cancelled');</script>";
    }
}
```

Снимка №40 – PHP условен оператор за отказване на преглед

В третата секция, пациентът може да види и предписаните бележки от лекаря и да ги отпечата при желание.

Welcome, Georgi Georgiev
Profile
Logout

Dashboard
Book Appointment
Appointment History
Prescriptions

Doctor Name	Appointment Date	Appointment Time	Diseases	Allergies	Prescriptions	Detailed Info
Konstantinova	2021-05-28	10:00:00	Flu	No	Paracetamol 2x a day and good resting.	View/Print

Снимка №41 – Екран със списък на отминали преглед

При натискане на бутона View/Print автоматично се отваря нов прозорец в браузъра, с възможности за преглед и печат. Екран на този прозорец е предстанен на снимка №42.

Generate Bill
1 / 1
100%

Thank you for using our services! Here is your details

Diagnose

Patient Name : Georgi Georgiev
Doctor Name : Konstantinova
Appointment Date : 2021-05-28
Appointment Time : 10:00:00
Disease : Flu
Allergies : No
Prescription : Paracetamol 2x a day and good resting.

Снимка №42 – Екран с предписана от лекар бележка

Формулярът за автоматичния генератор на pdf документ е следният:

```
<?php
if (isset($_GET["generate_bill"])) {
    require_once("../TCPDF/tcpdf.php");
    $obj_pdf = new TCPDF('P', PDF_UNIT, PDF_PAGE_FORMAT, true, 'UTF-8', false);
    $obj_pdf->SetCreator(PDF_CREATOR);
```

```

$objj_pdf->SetTitle("Generate Bill");
$objj_pdf->SetHeaderData('', '', PDF_HEADER_TITLE, PDF_HEADER_STRING);
$objj_pdf->SetHeaderFont(array(PDF_FONT_NAME_MAIN, '', PDF_FONT_SIZE_MAIN));
$objj_pdf->SetFooterFont(array(PDF_FONT_NAME_MAIN, '', PDF_FONT_SIZE_MAIN));
$objj_pdf->SetDefaultMonospacedFont('helvetica');
$objj_pdf->SetFooterMargin(PDF_MARGIN_FOOTER);
$objj_pdf->SetMargins(PDF_MARGIN_LEFT, '5', PDF_MARGIN_RIGHT);
$objj_pdf->SetPrintHeader(false);
$objj_pdf->SetPrintFooter(false);
$objj_pdf->SetAutoPageBreak(TRUE, 10);
$objj_pdf->SetFont('helvetica', '', 12);
$objj_pdf->AddPage();

$content = '';

$content .= '
    <br/>
    <h2 align ="center">Thank you for using our services! Here is your detail<br/>
    <h3 align ="center">Diagnose</h3>

';

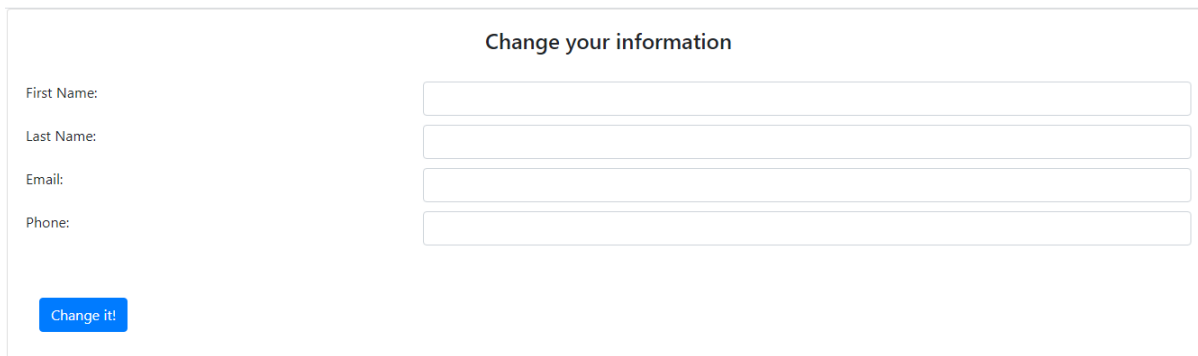
$content .= generate_bill();
$objj_pdf->writeHTML($content);
ob_end_clean();
$objj_pdf->Output("bill.pdf", 'I');
}

function get_specs()
{
    $con = mysqli_connect("localhost", "root", "", "myhmsdb");
    $query = mysqli_query($con, "select username,spec from doctb");
    $docarray = array();
    while ($row = mysqli_fetch_assoc($query)) {
        $docarray[] = $row;
    }
    return json_encode($docarray);
}

```

Снимка №43 – Автоматичен генератор на PDF документи

Пациентът също така може да редактира данни за своя акаунт чрез натискането на бутона Profile, намиращ се в Nav менюто (снимка №44).



Снимка №44 – Екран за промяна на потребителски данни

Формулярът използван за горния изглед:

```
<?php
session_start();

$con = mysqli_connect("localhost", "root", "", "myhmsdb");

$pid = $_SESSION['pid'];
$fname = $_SESSION['fname'];
$lname = $_SESSION['lname'];
$email = $_SESSION['email'];
$contact = $_SESSION['contact'];

if (isset($_POST['submit'])) {
    //variables

    $pid = $_SESSION['pid'];
    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $email = $_POST['email'];
    $contact = $_POST['contact'];

    $query = mysqli_query($con, "UPDATE patreg SET fname='$fname', lname='$lname', email='$email', contact='$contact' where pid='$pid'");

    if ($query) {
        echo "<script>alert('Profile succesfully updated');</script>";
    } else {
        echo "<script>alert('Unable to process your request. Please try again!');</script>";
    }
}
```

```
}
```

Снимка №45 – Формуляр и PHP код за промяна на потребителски данни

В случая правим заявка UPDATE към базата данни и по-конкретно към таблицата patreg, и променяме стойностите на съответните променливи.

3. Слой база данни

Този слой е отговорен за записите, тяхното зареждане и всички подобни операции. Той не е нужно да подлага на валидация данните, подадени му от бизнес слоя, но може да го уведомява за възникнали грешки при четене или запис. Този слой работи само с данните. В дипломната работа е използвана релационна база от данни MySQL. Базата се състои от общо 6 таблици. Импортирането на базата данни към проекта става посредством таблото phpmyadmin.

В дипломната работа са използвани полета от следните типове данни:

- Int – цяло число в интервала [-2147483648, 2147483647] или [0, 4294967295] като unsigned. Използват се за представяне на данни, представляващи цели числови стойности. В дипломната работа всички полета, които формират Primary Key са именно от този тип.
- Varchar – Полета от тип символен низ с променлива дължина. Използва се за тип на полета, в които ще се съхраняват символни низове, като например имена, описания, телефонни номера и т.н
- Date – Дата във формат YYYY-MM-DD. Минималната стойност е ‘1000-01-01’, а максималната ‘9999-12-31’.
- Time – поле от вида “YYYY-MM-DD HH:MM:SS”. Използва се за съхранение на информация за дата и час.

При създаването на базата данни са взети под внимание следните етапи:

- Определяне на целите на уеб сайта: определят се всички данни, които ще обработва системата.
- Определяне на таблиците: данните се разпределят в таблици след изясняване на целите на софтуера
- Определяне на полетата на таблиците: тип на данните, обхват, стойност по подразбиране и др.

Едно от най-големите предимства на една релационна база от данни е скоростта, с която тя може да сортира и извлича определена информация от масивите, които управлява. За да постигне това, MySQL и всички останали системи за управление на бази от данни използват оптимизирани механизми за съхранение на данни, наречени индекси. Индексите позволяват на системата да създаде представяне на колона, която може да претърсва със значителна скорост. В дефиницията на една таблица, колоната

или колоните, които декларираме като първичен ключ, автоматично ще бъдат индексирани. Имената на таблиците са добре подбрани, за по-голяма яснота. Имената на колоните в таблиците също достатъчно добре показват, каква информация съдържат. Типът на колоните е така избран, че да се въвеждат само такива данни, които отговарят на този тип, за да се избегнат нежелани грешки. За създаване на таблиците използваме SQL заявки.

Например – Таблична структура за таблица “Пациенти” изглежда така:

```
CREATE TABLE `patreg` (
  `pid` int(11) NOT NULL,
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL,
  `gender` varchar(10) NOT NULL,
  `email` varchar(30) NOT NULL,
  `contact` varchar(10) NOT NULL,
  `password` varchar(30) NOT NULL,
  `cpassword` varchar(30) NOT NULL
)
```

Снимка №46 – Структура на таблица “пациенти”

Ще бъдат разгледани някои от отделните таблици, съдържащи се в базата данни.

Таблицата на администратор:

Администраторът се характеризира с идентификационен номер, потребителско име и парола.

Колона	Тип	Ключ
adminID	int(11)	Primary
username	varchar(50)	
password	varchar(30)	

Снимка №47 – Структура на таблица “администратор”

Таблица за насрочване на час за преглед от пациента:

В таблицата за записване на часове се съдържат данни за идентификатор ID на самата таблица, идентификатор на пациента, неговото име и фамилия, пол, имейл, телефонен номер, обслужващият го лекар, дата и час на прегледа и два статуса, единия за пациента и другият за доктора.

Колона	Тип	Ключ
appointmentId	Int(11)	Primary
patientId	int(11)	
Fname	varchar(20)	
Iname	varchar(20)	
gender	varchar(10)	
email	varchar(30)	
contact	varchar(10)	
doctor	varchar(30)	
appdate	date	
appTime	time	
userStatus	int(5)	
doctorStatus	int(5)	

Снимка №48 – Структура на таблица “насрочване на час за преглед”

Глава 4. Перспективи, приноси и бъдещо развитие

1. Перспективи за бъдещо развитие

В конкретната дипломна работа се разработва уеб-базирана система. В основата е залегнала методологията за проектиране на софтуер на базата на модули. Системата е лесна за научаване и употреба и предлага редица ползотворни възможности. Продуктът е създаден, следвайки стандартен процес за разработка на софтуер. Първо се събират и анализират изискванията към системата. След това се проектира архитектурата с помощта на UML диаграми. Следва реализация, тестване и оценка на самата система. Перспективите за бъдещо развитие на представената уеб система за управление на болнично заведение в дипломната работа обхващат:

- Качване на системата на сървър
- Оптимизация
- Подобряване на дизайна
- Създаване на мобилно приложение, което да работи със системата на база под домейн
- Рефакторинг на сорс кода
- Добавяне на функционалност за платежи при необходимост
- Динамични известия при отлагане на час за преглед
- Имплементация на номериране на страници (pagination) за всички данни в list view.
- И други

Биха могли да се използват и строги научни критерии като интервюта, допитвания, анкети за оценка на използването на услугите, предлагани от представения софтуерен продукт.

2. Приноси

Изградена е уеб-базирана система за болнично заведение чрез направено проучване на проблематиката, разгледана от дипломанта, и чрез избора на подходящи технологии за реализация.

Основните приноси на представената дипломна работа са от научно-приложен и приложен характер и обхващат:

- Извличане на подходяща информация, тясно свързана със спецификите в дейността на лекари, администратори и пациенти.
- Изграждане на диаграми, описващи концептуален модел на софтуерната система
- Реализиране на интуитивен потребителски интерфейс
- Изграждане на база данни посредством SQL подходи

Заклучение

В началото на дипломната работа си поставихме основна цел, а именно – изграждане на уеб система за управление на дейности в здравно заведение. Конкретизирахме няколко съществени задачи, които следваше да бъдат разрешени. Първата задача бе да обосновем нуждата от създаване на система за здравно заведение, като се запознаем със сходни по функционалност системи. Следващата задача бе да анализираме най-важните функции и типове дейности на потребителския поток. След това бе извършен подбор на средства за реализация на софтуера за управление на дейности в здравно заведение. Чрез UML диаграми бе разработен и визуализиран част от модела на софтуерния продукт. Бе представена неговата структура и начини за използване. Най-съществената част бе етапът за разработка и оптимизиране на продукта. Целта бе постигната чрез създаване на софтуер, представляващ една уеб-базирана система. Систеმა, подпомагаща работата на административния и лекарския персонал, както и подобряваща процеса между пациент и доктор. Използвайки този софтуер, персоналът и потребителите на клиниката имат възможност да направят работата си по-ефективна и същевременно да спестяват време. Времето за изчакване на пациентите ще бъде неименуемо редуцирано, вследствие на това, че по-голямата част от процеса ще се извършва с помощта на компютър, а не на ръка. Освен това с използването на системата пациентите винаги ще са наясно със статута на своите прегледи.

Както със всяка софтуерна система, така и в разглежданата, съществува множество от функционалности, които все още не са създадени или въведени в експлоатация. В крайна сметка, най-важна е оценката на потребителя на софтуера, което до голяма степен ще определи и бъдещето развитие на системата.

Софтуерната индустрия се развива с изключително бързи темпове и софтуерните приложения стават все по-сложни и все по-критични. Качеството, макар и относително понятие, е най-значимата характеристика на софтуерния продукт. Качеството е съвкупност от много фактори, но двата най-важни са - способността на софтуера да изпълнява правилно своите функции при нормална работа и способността да се справя с непредвидени ситуации. Софтуерното инженерство непрестанно търси да подобри средства и техники за осигуряването им.

Можем да се надяваме, че функционалностите, които се прилагат в системата, ще спомогнат за разрешаването на проблемите в областта на здравеопазването.

Библиография

- [1] Уебсайт за обучение по кодиране W3Schools, <https://www.w3schools.com/sql/>, последно посещение 20.06.2020г.
- [2] Официален сайт на програмният език PHP, <http://php.net/docs.php>, последно посещение 24.06.2020г.
- [3] Официален сайт на Wikipedia, <https://en.wikipedia.org/wiki/PHP>, последно посещение 12.06.2020г.
- [4] Уебсайт за обучение по кодиране SoftUni, <https://softuni.bg/>, последно посещение 19.05.2020г.
- [5] Официален сайт на Unified Modeling Language (UML), <https://www.uml.org/>, последно посещение 27.06.2020г.
- [6] Официален сайт на WAMP Server, <https://www.wampserver.com/en/>, последно посещение 10.05.2020г.
- [7] Уебсайт и форум за помощ, <https://stackoverflow.com/>, последно посещение 20.06.2020г.