ПРИЛОЖЕНИЕ 1

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО БИТОВА ТЕХНИКА

гр. Пловдив, ул.„Иван Перпелиев“ №2; тел.:032/628 524

www.pgbt-plovdiv-bg.com; e-mail: pgbt2005@abv.bg

ПРОФЕСИЯ: „ТЕХНИК НА КП СИСТЕМИ“

СПЕЦИАЛНОСТ: „Компютърна Техника и Технологий“

ДИПЛОМЕН ПРОЕКТ

за придобиване на трета степен професионална квалификация

ТЕМА:

Разработване и оптимизиране на WEB сайт – фотоалбум на завършващите ученици от 12 В клас, специалност КТТ в ПГБТ гр. Пловдив 2022 / 2023 г.

Дипломант: Павлин Попов

Ръководител-консултант: *инж. Димитрина Цветкова*

Клас $12^{В}$

E-mail: pacopichmena123@gmail.com

1. Въведения

Като ученик прекарах последните пет години в създаване на спомени и развитие на образованието ми. Въпреки че имаше и добри, и лоши моменти, реших да ги уловя и съхраня на уникално място - уебсайт. Избрах да създам уебсайт поради многото му предимства пред други алтернативи. Като универсално достъпна платформа чрез интернет, уебсайтът може да бъде достъпен от всяко място и по всяко време от компютър или мобилно устройство.

Уебсайтът предоставя много начини, с които може да се покажат най-ценните ни спомени от училище. Включвайки мултимедийни елементи като изображения, видеоклипове и аудио, уебсайтът не само съживява спомените, но и пренася към дадения момент. Функционалността му позволява да бъде модифициран и персонализиран до последния детайл, за да съживи картината независимо дали е чрез използване на графики или анимации.

Уебсайтът се състои от 5 основни страници, всяка със свой собствен стил. Всяка от страниците има различна функционалност. Началната страница се фокусира върху класните ръководители. Те ще имат последната дума в края на завършването ни. Учениците имаха за задача всеки от тях да обобщи в едно изречение отнасящо се за годините ни в ПГБТ. Албумът съдържа снимки от 2018 до 2023. Делнична програма, която пресмята колко време остава до края на текущия час или до края на междучасието. Показва кой учебен час е сега и следващия час или междучасие. Последната страница се отнася за мен. В нея има кратко описание на проекта и мотивацията ми.

Уебсайтът ще продължи да се поддържа до следващата учебна година. След това ще се трансформира в pdf файл и на всеки ще му бъде предоставено копие.

С Ъ Д Ъ Р Ж А Н И Е

3.2. Видове

3.3. Широка съвместимост

3.4. Стратегии за съдържание

3.4.1. Ключови думи

3.4.2. Концепции на стратегията за съдържание

3.5. Дизайн

3.5.1. Ключови елементи

3.5.1.1. Оформление

3.5.1.2. Цветова схема

3.5.1.3. Типография

3.5.1.4. Навигация

3.5.1.5. Достъпност

3.5.1.6. Снимки и видеосъдържание

3.5.1.7. Мобилни устройства и преоразмеряване

3.5.1.8. Достъпност

3.5.2.

3.6. Потребителско изживяване

3.7. Оптимизация за търсачки

4. Създаване на уебсайта

4.1. Описание

4.2. Страници

4.2.1. Заглавна(Начало.html/index.html)

4.2.2. Ученици.html

4.2.3. Албум.html

4.2.3.1. Снимки

4.2.3.1.1. Компресор

2. Интернет

Интернет е голяма мрежа, която свързва компютри и устройства по целия Свят, позволявайки комуникация помежду си. Това става възможно чрез използването на протоколи, които са набор от правила, управляващи как устройствата комуникират едно с друго. Интернет ни дава достъп до огромно количество информация, което прави намирането на това, от което се нуждаем, по-бързо и по-лесно. Интернет улесни комуникацията ни с хора от всички краища на земното кълбо и ще продължава да разширява възможностите си.

3. Какво е Уебсайт?

Уебсайтът е съвкупност от уеб страници, достъпни чрез интернет. За да се създаде уебсайт, уеб разработчикът пише код, обикновено в HTML, CSS и JavaScript, който след това се поддържа в уеб сървър.

Уебсайтовете са вече необходим ресурс на обществото и предлагат множество приложения, които са от полза на всеки. Една важна употреба на уебсайтовете, която ще разгледаме в тази дипломна работа, е способността да съхраняват информация и да е показват по интерактивен и визуално красив начин.

3.1. Начин на достъп

Когато потребител иска да получи достъп до уебсайт, той въвежда URL адреса на уебсайта (Uniform Resource Locator) в своя уеб браузър. Браузърът изпраща заявка до сървъра на уебсайта за уеб страницата, която включва HTML код, изображения и друго съдържание, което съставлява страницата.

URL адресът на уебсайта всъщност е четим от човека начин за представяне на IP адрес, който е уникален идентификатор за устройство в интернет. Всяко

устройство в интернет има IP адрес, който се използва за маршрутизиране на пакети с данни между устройствата. Въпреки това, IP адресите са трудни за запомняне от хората, така че DNS (система за имена на домейни) е създадена, за да съпостави четливи имена на домейни към IP адреси.

DNS е разпределена система, която се състои от хиляди сървъри по целия свят. Когато потребител въведе URL адреса на уебсайт в браузъра си, браузърът изпраща заявка до DNS резолвер, който обикновено се предоставя от ISP (доставчик на интернет услуги) на потребителя. След това резолверът изпраща заявка до основен DNS сървър, който връща IP адреса на домейн сървъра от първо ниво за името на домейна на уебсайта (като .com или .org).

Резолверът изпраща заявка до домейн сървъра от най-високо ниво, който връща името на домейна на уебсайта. Сървъра за имена е отговорен за запис на IP адресите на устройствата, свързани с името на домейна на уебсайта.

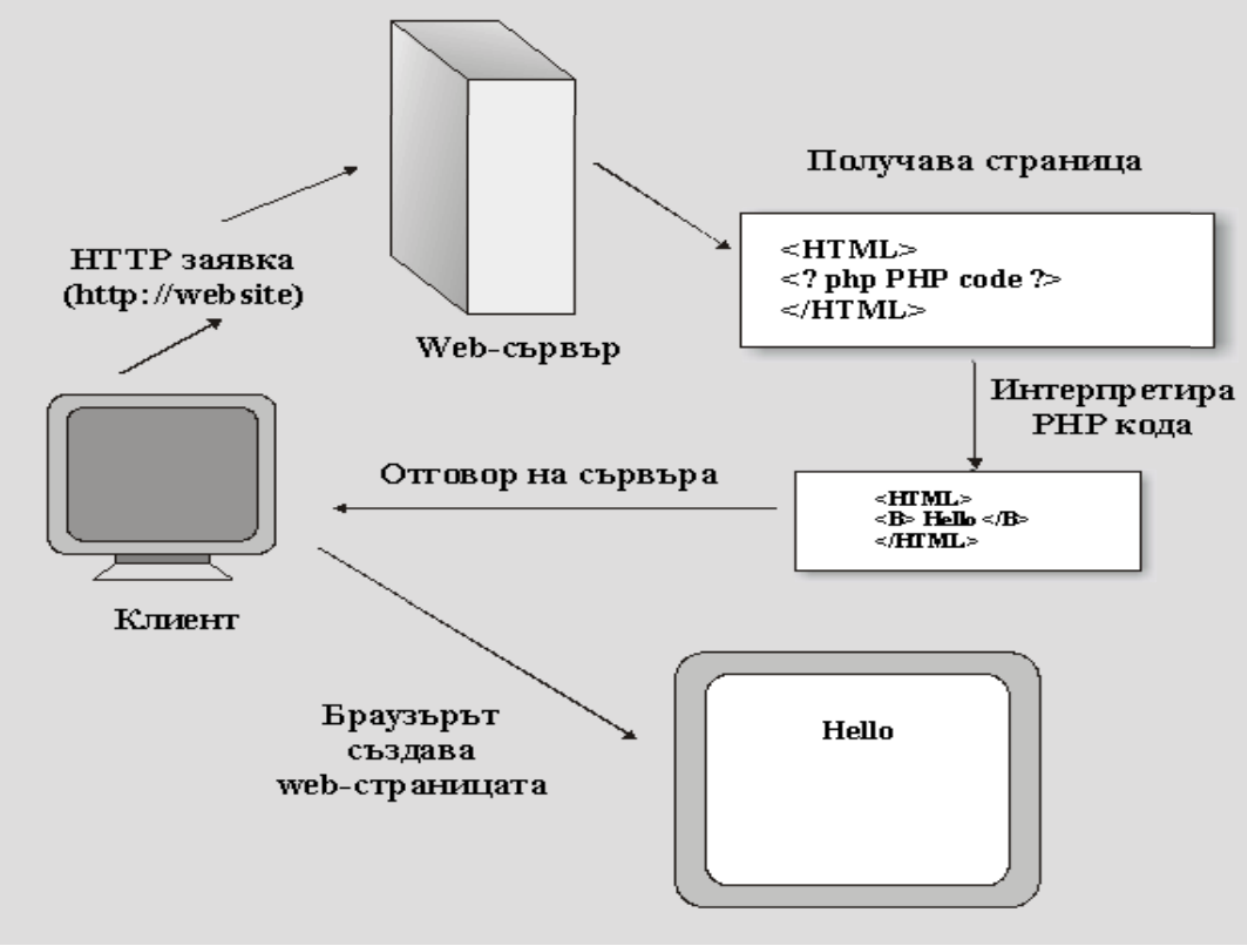
След като браузърът получи IP адреса на уебсайта, той изпраща заявка до сървъра за уеб страницата. Тази заявка включва информация за вида на използвания браузър, исканата страница и всички допълнителни параметри.

Сървърът получава заявката и започва да я обработва. Първо, сървърът може да провери дали исканата страница е в неговия кеш. Кеширането е процес на съхраняване на често достъпни данни в локално пространство за съхранение за по-бърз достъп. Ако исканата страница е в кеша, сървърът може да я върне веднага, без да се налага достъп до базата данни или други ресурси.

Ако страницата не е в кеша, сървърът извлича необходимите данни и генерира HTTP (Hypertext Transfer Protocol) отговор. Отговорът включва HTML код, изображения и друго съдържание, което съставлява уеб страницата, както и всякаква допълнителна информация, като например бисквитки.

Сървърът изпраща HTTP отговор обратно към браузъра на потребителя, който го получава и започва изобразяването на страницата. Браузърът използва HTML кода, за да покаже текста и изображенията на екрана.

След като страницата приключи изобразяването, потребителят може да взаимодейства със страницата, като кликва върху връзки, попълва формуляри или да извършва други действия. Когато потребителят щракне върху връзка, неговият браузър изпраща нова заявка до сървъра, който отговаря с подходящото съдържание за новата страница. Този процес продължава, докато потребителят напусне уебсайта или затвори своя браузър. По време на целия този процес сървърът на уебсайта непрекъснато комуникира с браузъра на потребителя, като обменя данни и изобразява съдържанието на страницата в реално време.



Едно допълнение към процеса са бисквитките. Те са малки файлове, които се съхраняват в браузъра и се използват за съхранение на информация за предпочитанията на потребителя, информация за влизане и други данни, които се използват за подобряване на работата на уебсайта. Бисквитките също могат да се използват за проследяване на активността на потребителя на уебсайта, въпреки че тази практика е предмет на разпоредби за поверителност.

3.2. Видове

Има голямо разнообразие от функции, които уебсайтовете могат да обслужват, като различни видове отговарят на различни нужди. Уебсайтовете могат да варират от лични блогове до онлайн магазини и от корпоративни сайтове до образователни ресурси. Съответно те могат да бъдат класифицирани в различни типове въз основа на тяхната предвидена функция и цел.

- Информационни: Много уебсайтове служат като източник на информация за определена тема, като новинарски уебсайтове или образователни уебсайтове. Тези уебсайтове обикновено включват статии, видеоклипове и други ресурси, които имат за цел да информират или образоват потребителя.
- Електронна търговия: Онлайн магазините са често срещан тип уебсайт, който позволява на потребителите да купуват продукти или услуги по интернет. Уебсайтовете за електронна търговия обикновено включват функция за пазарска количка, продуктови списъци и функционалност за плащане – eMag.bg.
- Социални: Социалните мрежи като Facebook, Twitter и Instagram позволяват на потребителите да се свърже с приятели и семейство чрез тях. Могат да споделят снимки и видеоклипове и да създават нови запознанства онлайн.

- Развлечения: Уебсайтове като YouTube, Netflix и Spotify предоставят развлекателно съдържание като видеоклипове, филми, телевизионни предавания и музика.
- Блогове: Блогове като WordPress и Medium позволяват на потребителя да създават и публикуват собствено съдържание в интернет, обикновено под формата на писмени статии.
- Портфолио: Много хора използват уебсайтовете като начин да покажат своята работа и умения пред потенциални работодатели или клиенти. Тези уебсайтове обикновено включват портфолио от работата на лицето, заедно с автобиография или друга информация за неговия опит.
- Правителствени: Държавните уебсайтове предоставят информация за държавни услуги, закони и разпоредби. Тези уебсайтове обикновено се поддържат от държавни агенции на местно, щатско или национално ниво.
- Дарения: Организациите с нестопанска цел често използват уебсайтове като начин за популяризиране на тяхната кауза и събиране на дарения от поддръжници. Тези уебсайтове обикновено включват информация за фонда, програмите и събитията на организацията.
- Автобиографични: Хората изразят себе си онлайн, обикновено чрез блог. Тези уебсайтове да включват информация за хобитата, интересите или личния живот на автора.

Важна подробност, която не бива да се изпуска, е вида на уебсайта според интерактивността му. Статичните показват едно и също съдържание на всеки потребител и не позволяват потребителско да въвеждат информация в базата данни. Динамичните, от друга страна, могат да генерират различно съдържание въз основа на въведени от потребителя данни

3.3. Широка съвместимост

Широката съвместимост на уебсайтове в интернет е критичен аспект, който позволява достъп до уеб съдържание на различни устройства и платформи. Уебсайтовете трябва да бъдат проектирани и разработени така, че да са съвместими с различни уеб браузъри, операционни системи, размери на екрана и интернет скорости.

За да постигнат съвместимост, уеб разработчиците използват уеб стандарти и най-добри практики, които гарантират, че уеб страниците се изобразяват правилно на всички устройства. Уеб стандартите са насоки за създаване на уеб страници, които гарантират съвместимост, достъпност и използваемост. Тези стандарти се поддържат от организации като World Wide Web Consortium (W3C) и Internet Engineering Task Force (IETF).

Удобният уеб дизайн е друг подход, използван от разработчиците за осигуряване на съвместимост. Тази техника включва проектиране на уеб страници, които могат автоматично да се адаптират към различни размери на екрана и различни устройства. Адаптивният уеб дизайн преоразмерява елементите на различните устройства, за да оптимизира страницата на различните устройства.

Обема на информация, който се пренася през интернет трябва да в границата на нормалното. В повечето случай се налага да се оптимизира информацията, за да забърза зареждането на страницата. Чрез компресиране на изображения, минимизиране на HTTP заявки и използване на мрежи за доставка на съдържание (CDN). CDN са мрежи от сървъри, които кешират съдържанието на уебсайтове на различни места по света, което улеснява достъпа до уебсайта от различни части на света.

3.4. Стратегии за съдържание

Стратегията за съдържание има голямо значение за създаването на уебсайта. Включил съм няколко дизайнерски идеи и методи в процеса на проектиране му.

В началото трябваше да направя дълбоко и обширно проучване за това какво ще бъде най-подходящо да се показва на самия уебсайт, така че да бъде в контекст с предстоящите събутия. Това ми помага да определя кои теми ще бъдат най-подходящи. Търсех начини да създам комбинация от най-доброто съдържание и най-добрия дизайн съчетани в едно. Това в крайна сметка се оказа от ключово значение за темата на проекта.

3.4.1. Ключови думи

Ключовите думи също са важен елемент от стратегията. Ключовите думи са думи или фрази, които посетителя обикновено използва, за да опишат информацията, която търсят. Те играят критична роля при насочването на потребителите към подходящо съдържание. Думите се ориентират около темата, така че посетителя да се навигира по страницата и като цяло в целия уебсайт. Тези думи са кратки и се намират от началото до края на всяка страница.

След като идентифицирахме нашите целеви ключови думи, важно е да ги използвате стратегически във нашето съдържание. Това не означава да напълним съдържанието си с възможно най-много ключови думи - това е тактика, известна като препълване с ключови думи, която всъщност ще да навреди на съдържанието и оформлението. Стремим се да използваме естествено количество ключови думи в съдържанието. Това означава да ги включим в заглавията, подзаглавията и в навигационното меню по начин, който има смисъл и се чете добре. Ако пък не можем да намерим нашите целеви ключови думи,

можем да използваме свързани ключови думи и синоними. Ключовите думи са само едно парче от пъзела за стратегия за съдържание.

3.4.2. Концепции

След като вече имаме установена цел с дадени критерии, започнаме да разработваме план за съдържание. Той включва комбинация от различни стратегически практики за съдържание, като се стремим да покрием широк набор от теми.

Една важна концепция на дизайна е визуалната йерархия, която приоритизира най-важната информация на страницата и ориентира потребителите. Постигаме визуална йерархия чрез използване на типография, разнообразни цветове и достатъчно разстояние между елементите.

Друг важен метод за проектиране, който използвам, е адаптивният дизайн. Той се адаптира към размера на екрана и ориентацията на устройството, осигурявайки оптимално преоразмеряване на елементите при навигиране. Тъй като все повече и повече хора използват мобилни устройства за сърфиране в интернет, адаптивният дизайн е основен метод за проектиране, за да се гарантира, че уебсайтът изглежда добре и функционира добре на всички устройства.

Минимализмът е една от популярните тенденции в планирането на дизайна, която е включена в повечето от нашите страници. Минималистичният дизайн използва прости форми, цветове и типография, за да създаде чист и непретрупан дизайн, който е лесен за четене.

Цветовото разнообразие е друга дизайнерска концепция, която е вложена. Цветовете могат да предизвикат определени емоции и настроения и разработчиците на уебсайтове могат да използват това в своя полза, като изберат

правилните цветове за своя уебсайт. Например, синьото често се свързва с доверие и надеждност, докато червеното се свързва със страст и вълнение.

Типографията е шрифтът, размерът и разстоянието на текста. Те са важни фактори, които трябва да имате предвид. Правилната типография прави уебсайта да изглежда професионален.

Добре планирана стратегия за съдържание, която взема предвид желаните ни резултати, значително подобрява ефективността на уебсайт като предоставя ясна информация, лесна навигация и стратегически етикети.

3.5. Дизайн

След като имаме изградена стратегия за съдържание, трябва да изготвим подходящ дизайн. Дизайна е много важна част от стратегията, защото визуализира съдържанието, така че да е лесен за възприема от читателя.

Дизайнът на уебсайт е критичен компонент на всеки уебсайт. Това включва планиране, създаване и актуализиране на оформлението, съдържанието и визуалните елементи на уебсайта. Добре проектираният уебсайт е от съществено значение за създаването на завладяващо и интуитивно потребителско изживяване, което насърчава посетителите да останат на сайта, да изследват съдържанието му и да предприемат действия.

3.5.1. Ключови елементи

Ключовите елементи

3.5.1.1. Оформление

Оформлението е първото нещо, което забелязваме, когато попаднем на уебсайта. Добре структурираното оформление помага се насочим към съдържанието, което

търсим по-лесно. За да създадат успешно оформление, дизайнерите трябва да обърнат внимание на няколко аспекта, включително разположението на съдържанието, използването на бяло пространство и цялостния поток на страницата. Може да се използва решетка или Wrapper, за да сме сигурни, че дизайнът е правилно подравнен и визуално балансиран. Важно е също уебсайтът да изглежда добре на всички устройства.

Едно от ключовите съображения при проектирането на оформлението на уебсайт е разположението на съдържанието. Ние се стремим да поставим най-важното съдържание на видно място в горната част на страницата или в центъра на страницата. Това може да включва основното послание на уебсайта, призив за действие или представен продукт. Поставянето на важно съдържание на видно място помага да се привлече вниманието на потребителя към него и ги насърчава да се ангажират с него.

Друг важен аспект на оформлението на уебсайта е бялото пространство, известно още като празни полета. Оставяме достатъчно бяло пространство между елементите, за да създадем визуален баланс и контраст между различните елементи на страницата. Освен това прави съдържанието по-лесно за четене и разбиране.

Потокът и йерархията също са важни съображения при проектирането на оформление на уебсайт. Стремим се да създадем ясна йерархия и поток от съдържание, който насочва погледа на потребителя от една секция към друга. Това помага да се създаде логична прогресия на информацията и позволява на потребителите лесно да разберат структурата на сайта. Йерархията може да бъде постигната чрез използване на различни размери и стилове на шрифта, цветови контраст и визуални знаци като стрелки или икони.

Често се използва Wrapper система, за да се създаде балансирано и структурирано оформление. Wrapper системата е рамка, която разделя страницата на колони и редове, което ни позволява да поставяме съдържание по последователен и организиран начин. Това също помага да се гарантира, че дизайнът и се адаптира към различни размери на екрана.

Визуалният баланс е важен аспект от оформлението на уебсайта. Стремим се да постигнем балансирано разпределение на визуални елементи, като изображения, текст и празно пространство.

3.5.1.2. Цветова схема

Изборът на правилната цветова схема е също важно за създаването на визуално привлекателен уебсайт. Цветовата схема трябва да се основава на идентичността на марката и целевата аудитория. Цветовете могат да предадат различни емоции и настроения, така че е важно да изберем цветове, които отговарят на желания тон на уебсайта или страницата. Една последователна цветова схема също може да помогне да създадем добро усещане.

Различните цветове могат да предизвикат различни емоции и да имат различно значение. За да изберем правилната цветова схема, първо вземаме предвид стила на уебсайта. Съобразяваме цветовата схема с емоцията или нещото, което искаме да покажем. Например в нашият случай началната ни страница има сив и златист оттенък, който цели да покаже колко скъпи са за нас класните ни ръководители. Като разбираме емоционалните и психологическите ефекти на различните цветове, ние избираме цветова схема, която е в съответствие с целите и посланията на уебсайта.

Контрастът на цветовете трябва да бъде балансиран. Високият контраст може да изпъкне важни елементи и да подобри четливостта, докато ниският контраст може да създаде по-фин и хармоничен ефект.

Цветовата хармония също е от съществено значение за един сплотен дизайн. Ние създаваме хармонична цветова схема, като използваме цветове, които се допълват взаимно. Можем да използваме цветове, които са един до друг в цветното колело, като аналогични цветове, или цветове, които са един срещу друг в цветното колело, като допълващи се цветове. Както в страницата ЗаМен.html цветовете се допълват един в друг и сменят позицията си.

И накрая, разглеждаме цялостната естетика на цветовата схема. Добре проектираната цветова схема трябва да бъде визуално привлекателна и подходяща за предназначението и аудиторията на уебсайта. Можем да експериментираме с различни цветови комбинации и да използваме инструменти като цветови палитри или инструменти за избор на цветове, за да си помогнем да изберем най-добрите цветове за нашия дизайн.

3.5.1.3. Типография

Когато проектираме уебсайта, избираме шрифт, който е подходящ за нашата целева група – училището, учителите, класа. Вземаме предвид фактори като четливост, мащабност и стил. Шрифт, който е твърде труден за четене, може да не изпъкне елементите достатъчно добре и да доведе до неудобство на читателите. Те ще предпочетат да не прекарват повече време и да ги накара да напуснат уебсайта. От друга страна, шрифт, който е твърде прост или скучен, може да не успее да привлече достатъчно внимание и да не успее да придаде нашето послание.

Също така вземаме предвид йерархията на съдържанието, когато избираме шрифт. Избираме шрифт, който прави разлика между различните видове съдържание на уебсайта, като заглавия, подзаглавия и основен текст. Използвайки различни размери и стилове на шрифта, ние създаваме ясна йерархия от информация, която помага на потребителите лесно да навигират и разбират съдържанието на уебсайта.

Обръщаме внимание и на разстоянието и височината на реда при избора на типография. Доброто разстояние между редовете и знаците подобрява четливостта прави съдържанието на уебсайта по-приветливо. Важно е да вземем предвид оформлението на уебсайта, когато избираме типография.

3.5.1.4. Навигация

Навигацията се отнася до откриването на съдържанието на уебсайта, за да помогне на потребителите да се навигират и пренасят до различни страници. Добре проектираната система подобрява ефективността.

Трябва да организираме съдържанието на уебсайта в логични категории и подкатегории, за да улесним потребителите да намерят информацията, от която се нуждаят. Трябва да се уверим, че система е лесна за използване и разбиране, дори за потребители, които не са запознати с уебсайта.

Използваме горния конти тур за меню с навигация в горната част на страницата. Там са представени страниците като при натиск те ще сменят цвета си в жълто и по този начин ще се активират като ви пренесат в страницата. Те са за цел да помогнем на потребителя да разбере какво ще намери, когато кликнат върху тях. При мобилната версия се използват икони, за да съберем връзките в оптимално пространство.

3.5.1.5. Практичност

Практичността се отнася до това колко лесно е на посетителя да взаимодейства с уебсайт. За да постигнем добра практичност, вземаме предвид различни фактори при проектирането на уебсайт.

Оптимизираме скоростта за зареждане, за да избегнем дългото чакане при зареждане на уебсайта. Компресираме изображенията и оптимизираме кода, за да осигурим кратко време за зареждане.

Фокусът ни върху доброто съдържание гарантира, че то е добре написано, лесно за четене и подходящо за конкретната страница. Използваме ясен език и не изключваме думи като жаргони или фрази, които биха загрозили съдържанието.

Тестваме функционалността на уебсайта, за да гарантираме, че всички функции работят правилно и идентифицираме и коригираме всички проблеми, преди уебсайтът да бъде готов за показване. Използваме анализи, за да проследяваме поведението на уебсайта и да го коригираме при нужда. Също така събираме обратна връзка от посетителите, за да идентифицираме евентуални проблеми и да подобрим приложимостта на уебсайта.

3.5.1.6. Снимки и видеосъдържание

Снимките и видеосъдържанието са голяма част от албума. Те имат за цел да покажат всичките спомени от нашето пребиваване в

гимназията. Снимките и видеоклиповете са сортирани според датата на създаване. Всички снимки и видеоклипове са компресиране и конвертирани в различни формати, за да се избегне бавното зареждане на страницата.

Снимките са компресиране чрез помощта на езика Python и различни библиотеки. Тъй като имаме голямо количество снимки и видеоклипове, необходимостта от конвертиране и компресиране е належаща. Създадохме програмен код, който да свърши всичката тази работа. Програмите работят в настоящата директория където се намират снимките и самата програма, която използваме.

4. Създаване на уебсайта

Уебсайтът е създаден посредством комбинация от софтуерни инструменти и програми за създаване. Те включват предимно текстови редактори, системи за актуализации, системи за управление на съдържанието, скриптови езици от страна на сървъра и от страна на клиента, уеб сървъри, инструменти за тестване и отстраняване на грешки и инструменти за внедряване.

4.1. Описание

Сайтът съдържа 5 главни страници и 5 подстраници, всяка която изпълнява различна роля. На всяка една от страниците има Header, който служи за навигация към различните страници. При натискане на текста, който всъщност е бутон, с името на страницата посетителят на уебсайта ще се пренесе в съответната страница, върху която е натиснал. Header е форматиран с шрифт "Open Sans" и

"Lobster", за да може да излгежда добре и лесен за навигация. В центъра на Header е изписано името на нашият клас, от ляво е текста(бутон) с навигация към началната страница, от дясно се намират и останалите страници "Ученици", "Албум", "Програма", "За мен". При посещение на всяка от страниците ще се активира текста и ще промени цвета си в златен цвят. Ще остане активен(текста(бутона)), докато не посети друга страница.

4.2. Страници

Страниците са 5 и всяка от тях изпълнява различна важна роля. Страниците са форматирана според идеята за създаването им и според стилизирането в цветовата гама на уебсайта.

Всяка страница съдържа Header и Footer.

4.2.1. Заглавна

Заглавната страница или началната страница е мястото където посетителят на уебсайта се натъква, след като въведе уеб адреса на уебсайта – 12vkttpgbt.ml. Началната ни страница е и нашата главна страница. Точно затова, когато посетителя потърси и въведе домейн адреса, той ще отвори началната страница – index.html. Тази страница е пълно копие на заглавната ни страница, която е с файлово име Начало.html.

Заглавната страница се дели на няколко части. Първата от които е място в самото начало на страницата където са разположени

няколко заглавия и кратък написан текст, който да изрази нашата благодарност към учителите. Класът, за който се отнася този уебсайт е изписан на едно от заглавията. Под него е випускът и след това специалността изучавана по време на престоя ни.

Спускаме се по-надолу и откриваме списък с автоматично сменящи се имена. Това е списък на поканените учители, които единодушно с класа решихме да поканим на нашия бал или нашето финално завършване или разделяне. Списъкът е интерактивен и всеки може да го прегледа от всяко мобилно устройство. Има и стрелки, които служат за навигация на сменяне на имената.

По нататък в страницата откриваме златно блестящ и лъскащ текст разположен в центъра на секцията. Тази секция е по-специална от другите, защото има две снимки. Те представляват нашите класни ръководители и техните снимки. Зад тях е създаден ефект на златен дъжд с помощта на инструменти и видео от платформата YouTube.com. Снимките имат и друга функционалност а тя е създаване на две нови секции или контейнери. Новите секции или контейнери се появяват с красива анимация при натискане на която и да е от снимките на двата ни класни ръководители. Новите контейнери съдържат снимка и текст под снимката, който цели да покаже финалната реч на нашите класни ръководители.

В края намираме златен текст изказващ датата на абитуриентския ни бал. Под него се намира таймер, който отброява дните, часовете, минутите и секундите оставащи до двадесет и пети май, когато всъщност е нашият бал. Зад текстовите елементи се намира canvas елемент, който създава ефект на празничност и дълбока потайност с графиките си. Фона е черен като най-специалното в този елемент са падащите звезди (пиксели) от програмата, която създадохме.

4.2.1.1. Оформление

4.2.1.2. Списък на учители

Така сме представили списък с имена, които се сменят периодично през 5 секунди.

```
<div class="carousel-inner" role="listbox">
<div class="align-center section-3-1" src="">
  <div class="sheet">
    <h1 class="title">инж.Димитрина Цветкова</h1>
  </div>
</div>
<div class="align-center section-3-2" src="">
  <div class="sheet">
    <h1 class="title">инж.Пенка Янева</h1>
  </div>
</div>
<div class="align-center section-3-3" src="">
  <div class="sheet">
    <h1 class="title">инж.Ивайло Иванов</h1>
  </div>
</div>
<div class="align-center section-3-4" src="">
  <div class="sheet">
```

```
    <h1 class="title">инж.Елена Кацарска</h1>
  </div>
</div>
<div class="align-center section-3-5" src="">
  <div class="sheet">
    <h1 class="title">инж.Петър Петров</h1>
  </div>
</div>
</div>
```

Списъкът се дефинира с помощта на контейнерен елемент с клас carousel-inner. Целта на този контейнерен елемент е да побере всички слайдове или елементи, които ще бъдат показани във списъка.

Всеки слайд или елемент се дефинира като div елемент с класовете align-center и section-3-N, където N е число от 1 до 5. Тези класове се използват от JavaScript кода за идентифициране на всеки слайд и визуализацията им.

Всеки слайд също съдържа div елемент с класовия лист, който се използва за целите на оформлението. Вътре в този елемент div има елемент h1 със заглавието на класа. Този елемент h1 съдържа името или заглавието на всеки слайд.

```
const carousel = document.querySelector('.u-carousel');
const carouselItems = document.querySelectorAll('.u-carousel-item');
const carouselInterval = 5000;
let carouselIndex = 0;

function showSlide(index)
{
  carouselItems.forEach(item => item.classList.remove('u-active'));
  carouselItems[index].classList.add('u-active');
}
```

```
function changeSlide(n)
{
  carouselIndex += n;
  if (carouselIndex < 0)
  {
    carouselIndex = carouselItems.length - 1;
  }
  else if (carouselIndex >= carouselItems.length)
  {
    carouselIndex = 0;
  }
  showSlide(carouselIndex);
}

function automaticSlide()
{
  changeSlide(1);
}

let interval = setInterval(automaticSlide, carouselInterval);

carousel.addEventListener('mouseover', () => clearInterval(interval));
carousel.addEventListener('mouseleave', () => interval =
setInterval(automaticSlide, carouselInterval));
document.querySelector('.u-carousel-arrow-left').addEventListener('click', ()
=>
{
  clearInterval(interval);
  changeSlide(-1);
});
document.querySelector('.u-carousel-arrow-right').addEventListener('click',
() =>
{
  clearInterval(interval);
  changeSlide(1);
});
```

Първо, кодът дефинира променлива currentItem, за да следи кой елемент от списъка се показва в момента. Инициализира се на 0, представлявайки първия елемент във въртележката.

След това кодът използва метода querySelector, за да избере елемента на въртележката и неговите дъщерни елементи. Той сам избира въртележката, като търси елемент с клас u-карусел. Той избира дъщерните елементи, като търси елементи с клас u-carousel-inner, който съдържа всички елементи на въртележката, и елементи с клас u-carousel-arrow, които са бутоните със стрелки наляво и надясно за навигация в въртележката.

Програмата добавя event listener към бутоните със стрелки наляво и надясно с помощта на метода addEventListener. Когато се щракне върху бутона със стрелка наляво, той извиква функцията showPreviousItem, която намалява променливата currentItem и извиква функцията showItem, за да покаже предишния елемент във въртележката. Когато се щракне върху бутона със стрелка надясно, той извиква функцията showNextItem, която увеличава променливата currentItem и извиква функцията showItem, за да покаже следващия елемент във въртележката.

Функцията showItem взема променливата currentItem и я използва, за да избере съответния елемент във въртележката с помощта на метода querySelectorAll. След това преминава през всички елементи от въртележката и добавя u-active класа към елемента, който съответства на променливата currentItem. Това прави този

елемент видим във въртележката, докато скрива всички останали елементи.

И накрая, програмата задава таймер с помощта на метода setInterval за автоматична промяна на елемента от въртележката на всеки 5 секунди. Функцията showNextItem се извиква във функцията на таймера, за да увеличи променливата currentItem и да покаже следващия елемент във въртележката. Идентификационният номер на таймера се записва в променлива, наречена таймер, така че да може да бъде спрян по-късно, ако е необходимо.

4.2.1.3. Ръководители

Класните ни ръководители имат специално място на нашия уебсайт. Мястото, на което се намират представлява падащ златен дъжд. Снимките са разположени централно на страницата и имат няколко приложения.

Златният дъжд е YouTube видео, което сме интегрирали и преоразмерили в страницата.

```
<div class="background-video expanded">
 <div class="embed">
  <iframe style="position: absolute;top: 0;left: 0;width: 100%;height: 100%;"
class="embed-responsive-item" src="(yt link)” data-autoplay="1"
frameborder="0">
  </iframe>
 </div>
</div>
```

Този HTML код създава видео с помощта на видеоклип в YouTube. с класа background-video задава фоновото видео да запълва целия прозорец за изглед, като използва CSS expanded клас.

В този div има друг елемент с класа embed, който се използва, за да направи видеото адаптивно. Елементът <iframe> е вграден в този div, който се използва за показване на видеоклипа в YouTube.

Атрибутът style в елемента <iframe> го позиционира да запълни цялата ширина и височина на родителския елемент. Атрибутът src на iframe указва URL адреса на видеоклипа в YouTube, който ще бъде възпроизведен като фонов видеоклип.

Атрибутът за автоматично пускане на данни е зададен на 1, което означава, че видеото ще започне да се възпроизвежда автоматично, когато страницата се зареди. И накрая, атрибутът frameborder е зададен на 0, което означава, че около видеото няма да се показва рамка.

```
<div class="u-expanded-width u-list u-list-1">
 <div class="repeater-1">
  <div id="cvetkova" class="container ..." data-image-width="2000" data-
image-height="1333">
   <div class="background-effect">
    <div class="background-effect-image image" data-image-width="301" data-
image-height="239">
    </div>
   </div>
   <div class="container-layout">
```

```
    <h3 class="font-lobster" style="...;">инж. Димитрина Цветкова</h3>
     <p class="font-merriweather" style="...;">"(Изречение от инж. Димитрина
Цветкова)</p>
   </div>
  </div>
  <div id="qneva" class="container ..." data-image-width="2000" data-image-
height="1333">
   <div class="u-background-effect">
    <div class="u-background-effect-image image" data-image-width="329" data-
image-height="373">
    </div>
   </div>
   <div class="container ...">
    <h3 class="...">инж. Пенка Янева</h3>
     <p class="font-merriweather;">"(Изречение от инж. Пенка Янева)"</p>
    </div>
  </div>
 </div>
</div>
```

Този код съдържа списък от два елемента, всеки представен от div елемент с клас repeater-item. Първият елемент е представен от div с id cvetkova, а вторият елемент е представен от div с id qneva.

Всеки елемент е затворен в div с класове u-expanded-width, u-list и u-list-1. Тези класове се използват за определяне на оформлението и стила на елементите от списъка.

Във всеки елемент от списъка има div с класове align-left-xs, border-3, border-palette-3-base, container-style, effect-hover-zoom, list-item и shading. Тези класове се използват за определяне на оформлението и стила на контейнера на елемента, включително ефектите на рамката, подравняването и засенчването.

В контейнера на елемента има два дъщерни елемента div. Първият дъщерен елемент div има клас background-effect и съдържа друг дъщерен елемент div с класове background-effect-image, expanded, image и image-1. Тези класове се използват за дефиниране на фоновия ефект и изображението, показвано във всеки елемент.

Вторият дъщерен елемент div на контейнера на елемента има класове container-layout, similar-container, valign-bottom-md, valign-bottom-sm, valign-bottom-xl и container-layout-1 или container-layout-2, в зависимост от артикула. Тези класове се използват за определяне на оформлението и позиционирането на съдържанието във всеки елемент.

Във втория съседен елемент div на всеки елемент има два дъщерни елемента: h3 заглавен елемент с класове custom-font, font-lobster, u-text, text-default и text-2 или text-4, в зависимост от елемента, и елемент p абзац с класове custom-font, font-merriweather, text, text-default и text-3 или text-5, в зависимост от артикула. Тези класове се използват за определяне на шрифта, цвета на текста и подравняването на текста на елементите на заглавието и абзаца.

Снимките имат функция, която при натискане върху която и да е от тях ще се покаже нова секция с снимка на класния ръководител и текст набран от него.

```
const Cvetkova = document.getElementById("cvetkova");
```

```
const Qneva = document.getElementById("qneva");
const imageContainer = document.getElementById("snimka1");
const imageContainer2 = document.getElementById("snimka2");

Cvetkova.addEventListener("click", function()
{
 toggleSection(imageContainer);
});
Qneva.addEventListener("click", function()
{
 toggleSection(imageContainer2);
});

function toggleSection(container)
{
 if (container.style.display === "none")
 {
  container.style.display = "block";
  container.style.opacity = "0";
  container.style.transition = "opacity 0.5s ease-in-out";
  setTimeout(() => {
   container.style.opacity = "1";
  }, 50);
 }
 else
 {
  container.style.opacity = "0";
  setTimeout(() => {
   container.style.display = "none";
  }, 500);
 }
}
```

Първо, кодът присвоява HTML елементи на променливи, използвайки техните идентификатори – по-специално два елемента с идентификатори "cvetkova" и "qneva" и други два елемента с идентификатори "snimka1" и "snimka2".

След това се добавят два event listener - единият към елемента cvetkova, а другият към елемента qneva. Когато се щракне върху някой от тези елементи, функцията toggleSection се извиква и предава съответния елемент на контейнера за изображение като аргумент.

Функцията toggleSection започва с оператор if, който проверява дали стилът на показване на контейнерния елемент е зададен на „none“. Ако е така, функцията задава стила на показване на "block", след това задава стила на прозрачност на 0 и прилага ефект на CSS преход, който променя свойството на прозрачност за период от 0,5 секунди. След 50 милисекунди функцията задава стила на прозрачност на 1.

Ако стилът на показване на елемента на контейнера не е зададен на „none“, което означава, че вече е видим, тогава функцията задава прозрачност на 0 и изчаква 500 милисекунди, преди да зададе стила на „none“.

4.2.1.4. Таймер

Последната секция представлява таймер разделен на дни, часове, минути и секунди, които отброяват времето оставащо до абитуриентския ни бал. Те са разположени централно и еднакво подравнени.

```
<div class="u-expanded-width u-list u-list-1">
  <div class="u-repeater u-repeater-1">
```

```
    <div class="center container ...">
      <div class="container-layout ...">
        <h3 id="days" class="font-ubuntu text-timer">010</h3>
        <h4 class="font-ubuntu text-days">Дни<br></h4>
      </div>
    </div>
    <div class="center container ...">
      <div class="container-layout ...">
        <h3 id="hours" class="font-ubuntu text-timer">00</h3>
        <h4 class="font-ubuntu text-hour">Часа</h4>
      </div>
    </div>
    <div class="center container ...">
      <div class="container-layout ...">
        <h3 id="minutes" class="font-ubuntu text-timer">00</h3>
        <h4 class="font-ubuntu text-min">Минути</h4>
      </div>
    </div>
    <div class="center container ...">
      <div class="container-layout ...">
        <h3 id="seconds" class="font-ubuntu text-timer">00</h3>
        <h4 class="font-ubuntu text-sec">Секунди</h4>
      </div>
    </div>
  </div>
</div>
```

Този дефинира списък от четири елемента, всеки от които съдържа два поделемента. Списъкът е стилизиран така, че да центрира съдържанието си, а използваният шрифт е Ubuntu. Всеки поделемент съдържа заглавен таг, съответно h3 и h4, който показва числова стойност и низ, който показва мерната единица за стойността.

Първият поделемент на всеки елемент от списъка съдържа атрибут id, който може да се използва за достъп и манипулиране на съдържанието му с JavaScript. Стойностите на атрибута на id са „дни“,

„часове“, „минути“ и „секунди“, което предполага, че кодът вероятно се използва за показване на таймер за обратно отброяване, който показва колко време остава до определено събитие.

```
var countDownDate = new Date("May 25, 2023 00:00:00").getTime();

var x = setInterval(function()
{
 var now = new Date().getTime();
 var distance = countDownDate - now;
 var days = Math.floor(distance / (1000 * 60 * 60 * 24));
 var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 *
60));
 var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
 var seconds = Math.floor((distance % (1000 * 60)) / 1000);

 document.getElementById("days").innerHTML = days;
 document.getElementById("hours").innerHTML = hours;
 document.getElementById("minutes").innerHTML = minutes;
 document.getElementById("seconds").innerHTML = seconds;
}, 1000);
```

Този код е на езика за програмиране JavaScript, който създава таймер за обратно отброяване. Той отброява до 25 май 2023 г. Целевата дата и час се задават с помощта на новия метод Date(), който приема аргумент от низ във формат „Месец Ден, Година Час:Минута:Секунда“. Това задава променливата countDownDate, която представлява целевата дата и час в милисекунди от 1 януари 1970 г.

След това обратното броене се инициира с помощта на метода setInterval(), който изпълнява функция многократно през определен интервал от време. В този случай функцията се изпълнява всяка

секунда (1000 милисекунди). Вътре във функцията текущата дата и час се получават с помощта на новия метод Date(), а оставащото време до целевата дата и час се изчислява в милисекунди с помощта на прости аритметични операции. След това оставащото време се преобразува в дни, часове, минути и секунди с помощта на метода Math.floor() и някои операции за деление и модул.

Таймерът за обратно отброяване се показва на страницата с помощта на инструмента innerHTML на метода document.getElementById(). Този метод извлича HTML елемента с определен ID атрибут и актуализира съдържанието му, за да покаже оставащите дни, часове, минути и секунди. Инструмента innerHTML се използва за модифициране на съдържанието в HTML елемент, а методът getElementById() се използва за достъп до HTML елемента със съответния ID атрибут. Чрез непрекъснато актуализиране на съдържанието на уеб страницата всяка секунда, таймерът за обратно отброяване динамично показва оставащото време до достигане на целевата дата и час.

4.2.2. Ученици

Целта на тази страница е да представи всеки ученик от класа с негова снимка и изречение или цитат, с което да опише престоя си в ПГБТ. Те имаха право да изберат своя снимка или да изкачат до бала, от който ще се сдобием с хубави снимки. Цитатът или изречението, което трябва да напишат трябва да е нещо, което те са усетили през престоя си в гимназията. Имат право да напишат всичко, което сметнат за правилно.

Снимката и цитатът всъщност представляват карта, която ученикът може да персонализира. Лицето на картата е самата снимка на ученика с името му в долния ляв ъгъл. При натискане на картата или снимката ще се създаде анимация на завъртане и гърба на картата ще се появи. Те също така имаха право персонализират картата си.

Вариантите бяха:

- Да се добави друга снимка по избор на ученика.
- Да се добави background снимка на гърба на картата.
- Да се добави бутон, който да включва звук, избран от ученика.
- Текста или цитата са напълно избираеми от всеки ученик.

Каквито идеи бяха предложени, такива бяха имплементирани. Никой не беше разочарован!

4.2.2.1. Оформление

4.2.2.2. Карти

Картите както споменахме по-горе са основен елемент в тази страница. Те представляват лице с снимка на ученика и текст с името на ученика, който се намира долу в ляво. И гръб с текст или цитат и свободно избираеми елементи по избор. Допълнителните елементи са:

- Снимка
- Видеоклип
- Текст
- Звук
- Анимация
- Допълнително форматиране на картата (по избор)
- Линк към социална медия (Facebook, Instagram, Github...)

```
<div class="flip-card">
  <div class="flip-card-inner">
    <div class="flip-card-front" data-image-width="770" data-image-
height="1031">
      <div class="...">
        <h3 class="...">Теодор Лебанов</h3>
      </div>
    </div>
    <div class="flip-card-back ...">
      <div class="...">
        <p class="text-2 text-back">"НАМИСЛИ..."</p>
        <a class="instagram-icon"...>
          <i class="fa fa-instagram instagram-icon" id="instagram"></i>
        </a>
        <a class="facebook-icon" ...>
          <i class="fa fa-facebook-square" id="facebook"></i>
        </a>
      </div>
    </div>
  </div>
</div>
```

Така изглежда структурата на нашата карта с предна и задна страна. Класът на flip-card се използва за определяне на външния контейнер на елемента на картата.

Вътре в картата има два основни елемента на разделителна способност, които са flip-card-inner и flip-card-front. Вътрешният елемент на flip-card-inner представлява контейнера както за предната, така и за задната страна на картата. Елементът flip-card-front представлява предната страна на картата, която съдържа съдържание.

Съдържанието на предната страна на картата е вложено вътре в елемент DIV с някои класове. Този div съдържа елемент на заглавие h3, който показва името на ученика, в случая „Теодор Лебанов“.

Елементът за връщане на flip-card-back представлява задната страна на картата, която първоначално е скрита от гледката. Задната страна съдържа DIV с някои класове и три вътрешни елемента. Първият елемент е на параграф P с името на класа „text-2 text-back“, което показва цитат. Останалите два елемента са <a> елементи, които съдържат икони. Тези елементи <a> се използват за свързване към профилите на социалните медии на лицето, представено от картата. Първият от елементите <a> се свързва към профил на Instagram, докато вторият елемент се свързва към профил във Facebook.

Функцията на картата се крие в преобръщането ѝ. За тази цел трябва да напишем код на JavaScript, който да преобръща картата при натискане върху нея. Програмата не е нужно да е на отделен файл, а може да е в HTML страницата.

```
$(".flip-card").click(function(event)
{
 if (event.target.tagName === 'BUTTON' || event.target.parentElement.tagName === 'A')
       {
       return;
       }
$(this).toggleClass("active");
});
```

Този код е функция на JavaScript, която използва библиотеката на jQuery, за да добави event listener за кликване към всички елементи с класа "flip-card". Когато потребителят кликне върху елемент с този клас, той превключва "active" клас за този елемент.

Функцията започва с избора на всички елементи с клас "flip-card", използвайки jQuery селектора "$(".flip-card")". След това методът .click() се използва за добавяне на event listener за кликване към всеки от тези избрани елементи. Тази функция приема събитие като параметър – щракване.

След това се използва оператор IF, за да се провери дали кликваният елемент е бутон или <a> таг. Това се прави, за да се предотврати превключването на кода на "active", когато потребителят щракне върху бутон или <a> таг вътре в елемента на флип-картата. Ако целевият елемент или неговият родителски елемент са бутон или <a> таг, кодът се връща и не изпълнява метода toggleClass.

В случай че ученикът иска да има бутон със звук ще трябва да създаден HTML и JavaScript код, за да изпълним неговата молба.

```
<button onclick="playAudio(event)" ontouchstart="playAudio(event)"
type="cardButton" class="button">...</button>
<audio class="myAudio">
  <source src="sound/audioKV.mp3" type="audio/mpeg">
</audio>
```

Елементът на бутона включва event listener OnClick, който включва функцията "playAudio" и event listener ontouchstart, който

също включва функцията "playAudio". Това означава, че когато бутонът се кликне или докосне, функцията "playAudio" ще бъде изпълнена.

Аудио елементът включва изходен елемент, който определя изходния файл за аудио файла, който ще се възпроизвежда. В този случай изходният файл е "sound/audiokv.mp3" и е от типа "Audio/MPEG".

```
$(".myButton").mousedown(function()
{
	var x = document.querySelector(".myAudio");
	x.currentTime = 0;
	x.play();
});

$(".myButton").mouseup(function()
{
	var x = document.querySelector(".myAudio");
	x.pause();
});

function playAudio(event)
{
	event.stopPropagation();
	var x = document.querySelector(".myAudio");
	x.currentTime = 0;
	x.play();
}
```

Този код е JavaScript код, който използва библиотеката на jQuery, за да добави event listener към елемент на HTML бутон с класа "cardButton". Когато бутонът се щракне и задържа надолу (събитие на

Mousedown), се възпроизвежда аудио файл. Когато бутонът се пусне (MouseUp Event), аудио файлът спира да се възпроизвежда.

Освен това кодът включва самостоятелна функция, наречена „playAudio“, която се извиква, когато бутонът се кликне и служи за същата цел като на Mousedown.

Първият слушател на събитието се добавя по метода JQuery .mouseDown(). Този метод взима функция като аргумент, който ще бъде изпълнен, когато се случи събитието на mousedown. Вътре в тази функция кодът избира аудио елемент на страницата, използвайки метода QuerySelector и клас ".myAudio". След това свойството на текущото време на този елемент е зададено на 0, което означава, че аудиото ще започне да възпроизвежда от самото начало. Накрая методът .play () се извиква на аудио елемента, който започва да възпроизвежда аудиото.

Вторият слушател на събитието се добавя по метода JQuery .mouseUp(). Този метод поема функция като аргумент, който ще бъде изпълнен, когато се случи събитието на мишката. Вътре в тази функция кодът избира същия аудио елемент, използвайки метода QuerySelector и клас ".myaudio". След това методът .pause () се извиква на аудио елемента, който спира да възпроизвежда аудиото.

Самостоятелната функция "playAudio" служи на същата цел като слушателя на събитията на mousedown. Когато бутонът се щракне, тази функция се извиква и тя избира аудио елемента, използвайки метода QuerySelector и клас ".myAudio". След това свойството на текущото време на този елемент е зададено на 0, което означава, че аудиото ще започне да възпроизвежда от самото начало. Накрая методът .play() се извиква на аудио елемента, който започва да възпроизвежда аудиото.

Функцията Event.StopPropagation() е включена във функцията playAudio, за да се предотврати разпространението на събитието в DOM дървото и задействане на други слушатели на събития, които могат да бъдат прикрепени към сродни елементи.

4.2.3. Албум

Както вече споменахме по-рано в дипломния проект снимките и видеосъдържанието се намират точно в нашия албум. Той се състои от 1 главна страница и 5 подстраници, които всяка от тях изпълнява различна роля, но с еднаква цел. Тук ще разгледаме как сме оформили съдържанието, конвертирали, сортирали и накрая показали снимките и видеосъдържанието.

Главната страница на албума служи като навигация за отделните години, в които сме разположили нашите снимки. На нея намираме навигационни бутони с текст годината, която искаме да посетим.

Нашите снимки и видеоклипове се нуждаят от специална обработка, за която ще използваме набор от трикове и инструменти. След обработката на снимките, те ще се покажат на страницата с помощта на библиотека наречена Nanogallery2. Тази библиотека е най-добрият вариант за нашия дипломен проект, тъй като предоставя възможности да се персонализира.

4.2.3.1. Оформление

4.2.3.2. Снимки

Снимките са важна част от нашия уебсайт, тъй като представляват уловени спомени от конкретен момент. Те са общо около 500 и са събрани от всякакви източници като: Facebook, Instagram, галерии и други. Снимките са разположени в библиотека за прожектиране на снимки. Събрани са от началото на престоя ни заедно с класа до абитуриентския бал. Те са групирани по класове, пример: 8 клас – 2018 до 2019 година. Те са официални и не толкова официални. Тъй като снимките са много и се налага да се направят няколко необходими оптимизации, които са важни за финалния

продукт. Разглеждаме ги в няколко стъпки, които следвахме докато създавахме албума.

Първата стъпка е да разделим снимките по дати и събития. Много е важно тази стъпка да се случва с необходимото време и внимание, тъй като е много лесно да объркаме времето и мястото на конкретната снимка, ако тя няма правилно зададена дата и час на създаване.

Втора стъпка е да изберем снимките, които искаме да конвертираме с цел оптимизация. От собствения си файлов формат ще се трансформират в WebP файлов формат. В сравнение с други формати на изображения, като JPEG и PNG, WebP предлага по-добро компресиране и по-малки размери на файловете, което помага за времето за зареждане на уеб страницата и намалява разхода на интернет трафик.

WebP също е силно съвместим с различни уеб браузъри, включително популярни като Google Chrome, Microsoft Edge и Mozilla Firefox. Това означава, че повечето потребители ще могат да преглеждат WebP изображения без проблеми.

Трета стъпка е да компресираме WebP файловете. WebP изображенията се компресират, за да се намали размерът на файла и да се улесни зареждането им. Компресирането на този вид формат не

е задължително. Но в нашия случай, тъй като имаме голямо количество снимки, които трябва също да могат да се заредят и от мобилни устройства с мобилни данни, ние трябва да намалим размера им възможно най-много. Размерът на файла ще се намали с около 20%, за да не доведе до известна загуба на качество на изображението.

Четвърта стъпка е сортирането на снимките. То се осъществява чрез програма, която ние създадохме и има за цел да преименува всички снимки от 001 до NNN според датата на създаването им.

Пета стъпка е да качим нашите снимки в HTML страницата ни. Това се осъществява с нашата среда за програмиране Visual Studio Code, като прибавяме всяка снимка на отделен ред и според номерацията ѝ в тага, който е всъщност нашата галерия

4.2.3.2.1. Конвертор

Втората стъпка е да преместим всички снимки, които искаме да конвертираме в една папка по избор. За да конвертираме снимките по бърз и ефективен начин ще трябва да създадем програма, която да спести много от нашето време загубено в конвертиране с използването на онлайн конвертори. Програмата е на езика Python и използва няколко библиотеки за тази цел.

```
import os
import sys
```

```
import logging
from PIL import Image

logging.basicConfig(level=logging.ERROR)

if len(sys.argv) > 1:
    dir_path = sys.argv[1]
else:
    dir_path = os.path.dirname(os.path.abspath(__file__))

try:
    os.chdir(dir_path)
except FileNotFoundError:
    logging.error('Directory not found: {}'.format(dir_path))
    sys.exit(1)

img_folder = os.getcwd()

output_folder = os.path.join(img_folder, 'webp_images')
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

for filename in os.listdir(img_folder):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        try:
            with Image.open(filename) as im:
                output_filename = os.path.splitext(filename)[0] + '.webp'
                output_path = os.path.join(output_folder, output_filename)
                im.save(output_path, 'webp')
        except OSError:
            logging.error('Failed converting files: {}'.format(filename))

print('All JPG and PNG files have been converted and placed in: ' +
output_folder)
```

Програмата започва с импортиране на необходимите модули: os, sys, logging и PIL. Os модулът предоставя начин за използване на функционалност, зависима от операционната система, като например промяна на текущата работна директория. Модулът sys осигурява

достъп до някои променливи, използвани или поддържани от преводача и до функции, които взаимодействат силно с преводача, като достъп до аргументи на командния ред. Модулът за регистриране осигурява гъвкаво регистриране на съобщения към различни изходи. PIL модулът (Python Imaging Library) добавя поддръжка за отваряне, манипулиране и запазване на много различни формати на файлове с изображения.

Задава се нивото за регистрация на грешки, използвайки метода basicConfig от модула за регистриране. Това означава, че ще бъдат регистрирани само съобщения за грешки.

След това програмата проверява дали е предоставен път на директория като аргумент на команден ред, използвайки списъка sys.argv. Ако е предоставен път на директория, той е присвоен на променливата dir_path. Ако не е предоставен път на директорията, ще използва методите os.path.dirname и os.path.abspath, за да получи абсолютния път на директорията, където се намира скриптът и го присвоява на променливата dir_path.

Нататък променя текущата работна директория в посочения път на директорията, използвайки метода os.chdir. Ако посочената директория не съществува, FileNotFoundError се призовава и се хваща от except блок. В този случай се регистрира съобщение за

грешка по метода logging.error и скриптът излиза с код за грешка 1, използвайки метода sys.exit.

Ако не възникне грешка при промяна на текущата работна директория, скриптът продължава, като получи текущата работна директория, използвайки метода os.getcwd и го присвоява на променливата img_folder. След това скриптът създава нова поддиректория, наречена webp_images в текущата работна директория, използвайки методите os.path.join, os.makedirs и os.path.exists.

След това скриптът повтаря всички файлове в текущата работна директория, използвайки метода os.listdir. За всеки файл той проверява дали името му завършва с .jpg или .png. Ако го направи, той го отваря по метода image.open от PIL модула. След това скриптът създава ново име на файл, като замества разширението на файла с .webp. Създава изходен път, като се присъедини към пътя на изходната папка с това ново име на файл, използвайки метода os.path.join.

След това скриптът запазва това изображение във формат .webp на този изходен път, използвайки метода .save() от класа на изображението на PIL. Ако възникне грешка при отваряне или запазване на изображение (като OSError), тя се хваща от изключен

блок и съобщение за грешка се регистрира с помощта на logging.error.

След като всички изображения са обработени, се отпечатва съобщение, което показва, че всички .jpg и .png файлове са преобразувани и поставени в поддиректория, наречена „webp_images“.

4.2.3.2.2. Компресия

Следваща стъпка от обработката на снимките е тяхната компресия. За да може размерът на снимките да не бъде толкова голям, те трябва да се компресират и в същото време да не губят качеството на изображението. Отново ще използваме езика за програмиране Python, за да създадем програма, която да улесни нашата работа по снимките. Процесът с тази програма е автоматичен и е необходимо програма да се намира в директорията на снимките. Важно нещо обаче, което трябва да кажем е, че програмата ще конвертира .webp файлов формат. Той е много по-лек и е по-добрата опция, вместо да конвертираме традиционните файлови снимкови файлови формати като .png или .jpg.

```
import os
from PIL import Image

script_dir = os.path.dirname(os.path.abspath(__file__))

compressed_dir = os.path.join(script_dir, 'compressed')
```

```
if not os.path.exists(compressed_dir):
    os.mkdir(compressed_dir)

for filename in os.listdir(script_dir):
    if filename.endswith('.webp'):
        with Image.open(filename) as img:
            quality = 70
            output_path = os.path.join(compressed_dir, filename)
            img.save(output_path, 'webp', quality=quality)

print("Image compression is complete!")
```

Този код на Python компресира всички WebP изображения в директорията, където се намира програмата. Тя използва os модула, за да получи пътя на директорията на програмата на Python, като извика os.path.abspath(__file__), за да получи абсолютния път на файла на скрипта, и os.path.dirname(), за да получи пътя на директорията. Променливата script_dir съдържа този път на директория.

След това кодът създава нова директория с име „compressed“ в директорията script_dir, ако тя все още не съществува. Това е мястото, където ще бъдат запазени компресираните WebP изображения. Функцията os.path.join() се използва за създаване на път към „компресираната“ директория чрез свързване на променливата script_dir с името на директорията „компресирана“. Функцията os.mkdir() се използва за създаване на директорията, ако тя не съществува.

След това кодът преминава през всички файлове в директорията script_dir с помощта на функцията os.listdir(). За всеки файл кодът

проверява дали името на файла завършва с разширението „.webp“, като използва метода endswith(). Ако името на файла отговаря на това условие, кодът отваря файла с помощта на метода Image.open() на PIL модула и чете съдържанието му.

След това кодът компресира отвореното изображение с помощта на метода img.save(), като посочва качеството на компресията с помощта на параметъра за качество. В този случай качеството е зададено на 70. След това компресираното изображение се записва в директорията „compressed“. Функцията os.path.join() се използва за създаване на път към изходния файл в директорията „compressed“ чрез свързване на променливата compressed_dir с името на файла.

Накрая кодът отпечатва съобщение, което показва, че компресията на изображението е завършена. Това съобщение се показва, след като всички WebP файлове в директорията бъдат обработени.

4.2.3.2.3. Сортиране

Последната програма, която ще създадем е програма, която да намира всички .webp файлове, които да сканира и подреди по дата на създаване. Този процес е нужен да се осъществи, за да се открият и подредят заснетите моменти в хронологичен ред.

```
import os
import datetime

dir_path = os.path.dirname(os.path.realpath(__file__))
```

```
os.chdir(dir_path)
webp_files = [f for f in os.listdir() if f.endswith('.webp')]
webp_files.sort(key=lambda x: os.path.getctime(x))

for i, filename in enumerate(webp_files):
    new_name = f"{i+1:03d}.webp"
    os.rename(filename, new_name)
```

Този код на Python преименува всички WebP изображения в директорията, където се намира програмата, в цифров ред въз основа на времето на тяхното създаване. Първо използва модула os, за да получи пътя на директорията, като извиква os.path.abspath(__file__), за да получи абсолютния път на файла, и os.path.dirname(), за да получи пътя на директорията. Променливата dir_path съдържа този път на директория.

Функцията os.chdir() се използва за промяна на текущата работна директория на dir_path. Това гарантира, че последващото извикване на функцията os.listdir() работи в правилната директория.

След това кодът създава списък на всички файлове в директорията с разширение „.webp“. Това става с помощта на функцията os.listdir() и филтриране на резултатите с метода endswith(). Списъкът webp_files се сортира въз основа на времето на създаване на всеки файл. Това се прави с помощта на функцията os.path.getctime() като ключ за сортиране в функцията.

Програмата преминава през сортирания списък webp_files, използвайки for цикъл с функцията enumerate(), за да следи индекса

на всеки файл. За всеки файл се генерира ново име на файл с помощта на f-низ с цяло число с нула, представляващо индекса плюс едно. Това гарантира, че имената на файловете са номерирани в цифров ред с водещи нули. Например, първият файл е наречен "001.webp", вторият файл е наречен "002.webp" и т.н.

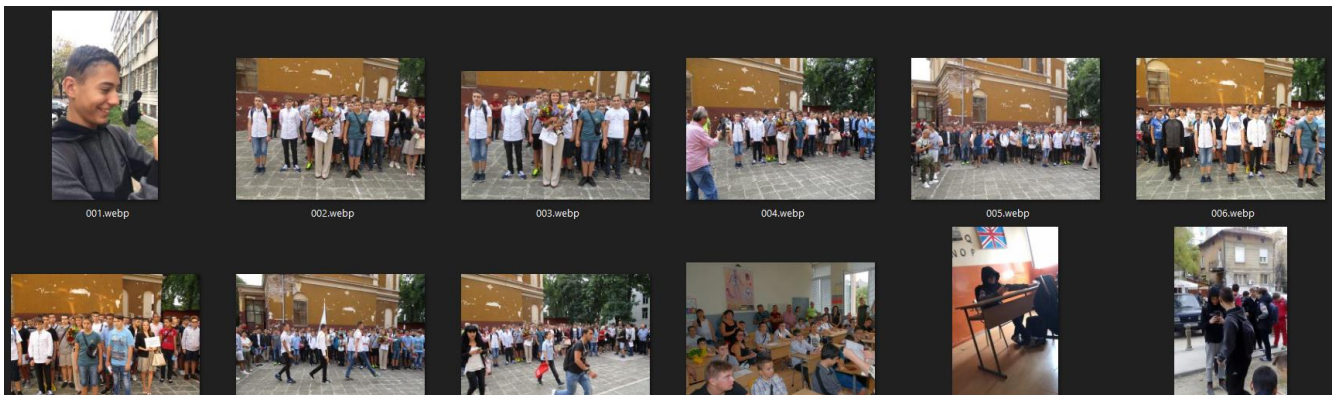
И накрая, програмата преименува всеки файл с помощта на функцията os.rename(). Тази функция приема два аргумента: оригиналното име на файла и новото име на файл. В този случай променливата име на файл съдържа оригиналното име на файл, а променливата new_name съдържа новото име на файл. След като всички файлове бъдат преименувани, числовият ред на имената на файловете съответства на времето им на създаване и успешно се преименуват всички WebP файлове в директорията.

4.2.3.2.4. Ръчно преглеждане за грешки и преименуване

Оказа се, че снимките, макар и от надеждни източници, са разбъркани и в голямата си част нямаше как да ги подредим по дата на създаване. Затова се наложи да прегледаме всяка снимка, след като вече сме ги компресирали, конвертирали и сортирали. Всъщност това не се оказа загуба на време, защото така или иначе не извършихме по-горните процеси ръчно. Снимките, които сметнахме, че са по-напред от останалите, ги преименувахме с номерация от 1 до N. Това в крайна

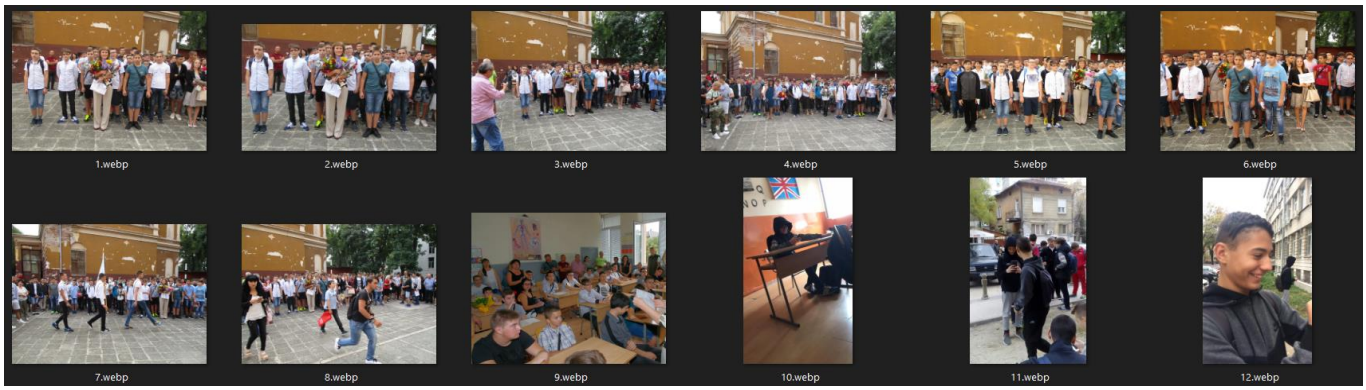
сметка ни позволи да обърнем по-голямо внимание на добрите кадри и ги изместихме по-напред.

Преди (4.2.3.2.4. фиг. 1):



4.2.3.1.4. фиг. 1

След (4.2.3.2.4. фиг. 2):



4.2.3.1.4. фиг. 2

4.2.3.2.5. Импортиране

Импортираме или качваме снимките в HTML страниците ни с помощта на още една програма на Python, която създадохме за да

автоматизираме процеса на въвеждане. Целта на програмата е да въведе всички необходими думички и ключови думи и елементи, с които да импортираме снимките в страницата по бърз и лесен начин.

```
with open('numbers.txt', 'w') as f:
    for i in range(1, 126):
        f.write(f'<a href="album/12/sporten/{i}.webp"></a>\n')
```

Този код на Python генерира файл с име numbers.txt със 125 реда HTML код, всеки от които съдържа връзка към файл с изображение на WebP.

Кодът започва с отваряне на нов файл с име numbers.txt в режим на запис с помощта на функцията open(). Файлът се отваря в команда with, която гарантира, че файлът се затваря автоматично при излизане от блока. Файловият обект се присвоява на променливата f.

Цикълът for итерира в диапазон от числа от 1 до 125, като използва функцията range(). За всяка итерация цикълът генерира ред от HTML код, който съдържа връзка към файл с изображение на WebP. Това се прави с помощта на f-низ, който съдържа форматиран HTML anchor таг с атрибута href, зададен на URL адрес, който сочи към WebP файл с изображение. Променливата i се използва за генериране на името на файла с изображение, който е включен в URL адреса.

Генерираният HTML код за всяка итерация се записва във файла numbers.txt с помощта на функцията f.write(). Кодът записва всеки ред

във файла със знак за нов ред в края, така че всеки ред е на отделен ред във файла.

След като цикълът завърши всички итерации, блокът with излиза и файлът се затваря автоматично.

Като цяло този код е кратък, но доста полезен инструмент за нашия случай.

Резултата от програмата намираме в numbers.txt (4.2.3.2.5 фиг. 1).

```
<a href="album/12/sporten/1.webp"></a>
<a href="album/12/sporten/2.webp"></a>
<a href="album/12/sporten/3.webp"></a>
<a href="album/12/sporten/4.webp"></a>
<a href="album/12/sporten/5.webp"></a>
<a href="album/12/sporten/6.webp"></a>
<a href="album/12/sporten/7.webp"></a>
<a href="album/12/sporten/8.webp"></a>
<a href="album/12/sporten/9.webp"></a>
<a href="album/12/sporten/10.webp"></a>
<a href="album/12/sporten/11.webp"></a>
<a href="album/12/sporten/12.webp"></a>
<a href="album/12/sporten/13.webp"></a>
<a href="album/12/sporten/14.webp"></a>
<a href="album/12/sporten/15.webp"></a>
<a href="album/12/sporten/16.webp"></a>
<a href="album/12/sporten/17.webp"></a>
<a href="album/12/sporten/18.webp"></a>
<a href="album/12/sporten/19.webp"></a>
<a href="album/12/sporten/20.webp"></a>
<a href="album/12/sporten/21.webp"></a>
<a href="album/12/sporten/22.webp"></a>
<a href="album/12/sporten/23.webp"></a>
<a href="album/12/sporten/24.webp"></a>
<a href="album/12/sporten/25.webp"></a>
<a href="album/12/sporten/26.webp"></a>
<a href="album/12/sporten/27.webp"></a>
<a href="album/12/sporten/28.webp"></a>
<a href="album/12/sporten/29.webp"></a>
<a href="album/12/sporten/30.webp"></a>
<a href="album/12/sporten/31.webp"></a>
<a href="album/12/sporten/32.webp"></a>
<a href="album/12/sporten/33.webp"></a>
```

4.2.3.2.5 фиг. 1

4.2.4. Програма

Тази страница представлява делничната програма на всеки ученик от 12 В клас. Тя се разглежда със снимки и система за пресмятане на текущите часове. Между часовете се намират междучасията, които също се пресмятат от системата. На първия елемент се показва текущия час или междучасие. Втория елемент представлява времетраенето до края на съответния елемент (час или междучасие). И третия елемент е с две думи – какво следва. Часовете се пресмятат и изреждат автоматично от програмата или системата, която създадохме. Времетраенето се показва в минути.

4.2.4.1. Оформление

Тази страница има за цел да представи програмата по красив и достъпен начин, затова създадохме няколко приложения.

Страницата е подредена както по предназначение, така и по дизайн. Всяка важна част е отделена от друга, за да може да се открояят.

4.2.4.1.1. Секция 1

```
<section class="u-align-center u-clearfix u-custom-color-6 u-section-1"
src="" id="sec-fb19">
 <div class="u-clearfix u-sheet u-valign-middle u-sheet-1">
```

```
  <h1 class="u-custom-font u-font-lobster u-text u-text-default u-title u-
text-1">Текущи часове</h1>
  <img class="u-image u-image-contain u-image-default u-image-1"
src="images/image1.png" alt="" data-image-width="400" data-image-
height="378">
 </div>
</section>
```

Този HTML код е ‘section’ елемент с четири CSS класа, приложени към него: u-align-center, u-clearfix, u-custom-color-6 и u-section-1.

Класът u-align-center определя свойството text-align да бъде центрирано, което означава, че всеки текст в елемента на раздела ще бъде центриран.

Класът „u-clearfix“ дефинира свойството display да бъде table, което означава, че всички плаващи елементи в елемента на раздела ще бъдат изчистени.

Класът „u-custom-color-6“ дефинира две свойства: цвят и цвят на фона. Свойството color задава цвета на текста на бял (#ffffff), а свойството background-color задава цвета на фона на тъмно синьо (#17233a).

Класът u-section-1 дефинира свойството background-image да бъде none, което означава, че в елемента section няма да се показва фоново изображение. Също така определя свойството background-color да бъде със същия тъмносин цвят като u-custom-color-6 клас.

4.2.4.1.2. Секция 2

```
<section class="u-align-center u-clearfix u-custom-color-6 u-section-2"
src="" id="carousel_7680">
 <div class="u-clearfix u-sheet u-sheet-1">
  <h1 class="u-custom-font u-font-lato u-text u-title u-text-1">Текущ час
  </h1>
  <h1 id="chasNow" class="u-custom-font u-font-ubuntu u-text u-text-2">часСега
  </h1>
  <h1 class="u-align-center-xs u-custom-font u-font-lato u-text u-title u-
text-3">Остават
  </h1>
  <h1 id="chasRemain" class="u-custom-font u-font-ubuntu u-text u-text-
4">часОстават
  </h1>
  <h1 class="u-custom-font u-font-lato u-text u-title u-text-5">Следва
  </h1>
  <h1 id="chasNext" class="u-custom-font u-font-ubuntu u-text u-text-
6">часСледващ
  </h1>
 </div>
</section>
```

Елементът section в HTML кода има атрибут class, който дефинира няколко имена на класове: u-align-center, u-clearfix, u-custom-color-6 и u-section-2. Тези класове се използват за прилагане на различни стилове към елемента на раздела и неговите близки елементи.

Вътре в елемента section има div елемент с имена на класове u-clearfix и u-sheet-1. Този елемент div съдържа няколко елемента h1 с различни имена на класове, които се използват за показване на различни видове текст на страницата.

Първият h1 елемент с клас u-custom-font u-font-lato u-text u-title u-text-1 се използва за показване на текста "Текущ час" на страницата. Класовете u-custom-font и u-font-lato определят персонализирано

семейство шрифтове за текста, докато класът u-text дефинира основни текстови стилове като размер на шрифта, височина на реда и цвят. Класът u-title добавя малко допълнителен стил към текста, за да се открои повече като заглавие. И накрая, класът u-text-1 се използва, за да посочи, че това е първият екземпляр на елемент с клас u-text в рамките на елемента section.

Вторият елемент h1 с клас u-custom-font u-font-ubuntu u-text u-text-2 се използва за показване на текущия час на страницата. Класовете u-custom-font и u-font-ubuntu дефинират персонализирано семейство шрифтове за текста, докато класът u-text дефинира основни текстови стилове. Атрибутът id на този елемент е зададен на chasNow, което позволява на JavaScript кода да актуализира динамично текста на този елемент.

Третият елемент h1 с клас u-align-center-xs u-custom-font u-font-lato u-text u-title u-text-3 се използва за показване на текста "Остават" на страницата. Класът u-align-center-xs центрира текста в елемента на малки екрани, докато класовете u-custom-font и u-font-lato определят персонализирано семейство шрифтове за текста. Класът u-text дефинира основни текстови стилове, докато класът u-title добавя допълнителен стил към текста, за да се открои повече като заглавие. И накрая, класът u-text-3 се използва, за да посочи, че това е третият екземпляр на елемент с клас u-text в рамките на елемента section.

Четвъртият елемент h1 с клас u-custom-font u-font-ubuntu u-text u-text-4 се използва за показване на оставащото време на страницата. Класовете u-custom-font и u-font-ubuntu дефинират персонализирано семейство шрифтове за текста, докато класът u-text дефинира основни текстови стилове. Атрибутът id на този елемент е зададен на chasRemain, което позволява на JavaScript кода да актуализира динамично текста на този елемент.

4.2.4.2. Дневна програма - система

Дневната програма представлява часовете от програмата в училище, като към тях прибавяме и междучасията.

За целта трябва да създадем програма, която да извършва тези операции.

```
function checkTime() {
      const chasNowElem = document.getElementById("chasNow");
      const chasRemainElem = document.getElementById("chasRemain");
      const chasNextElem = document.getElementById("chasNext");
      const currentTime = DateTime.local();
      let nextState = '';
      let isEventFound = false;

      for (const { start, end, state } of schedule)
      {
        const startDateTime = DateTime.local().set({ hour:
start.split(':')[0], minute: start.split(':')[1] });
        const endDateTime = DateTime.local().set({ hour: end.split(':')[0],
minute: end.split(':')[1] });
        if (currentTime >= startDateTime && currentTime <= endDateTime)
        {
          const diffTime = endDateTime.diff(currentTime, ["hours", "minutes",
"seconds"]);
          chasNowElem.innerHTML = state;
          chasRemainElem.innerHTML = diffTime.values.hours === 0 ?
`${diffTime.values.minutes} минути` : `${diffTime.values.hours} часа и
${diffTime.values.minutes} минути`;
          nextState = state;
          isEventFound = true;
          break;
        }
      }
      if (!isEventFound)
      {
        if (currentTime.hour <= 13)
```

```
        {
          const { state: firstEventState, start: firstEventStart } =
schedule[0];
          const nextState = schedule[2].state;
          const startTime = DateTime.local().set({ hour:
firstEventStart.split(':')[0], minute: firstEventStart.split(':')[1] });
          const razlika = startTime.diff(currentTime, ["hours", "minutes",
"seconds"]);
          const hoursText = razlika.values.hours === 1 ? 'час' : 'часа';
          chasNowElem.innerHTML = `${firstEventState} след
${razlika.values.hours} ${hoursText} и ${razlika.values.minutes} минути`;
          chasRemainElem.innerHTML = `${razlika.values.hours} ${hoursText} и
${razlika.values.minutes} минути`;
          nextState = schedule[2].state;
        }
        else
        {
          chasNowElem.innerHTML = 'Училището свърши';
          chasRemainElem.innerHTML = 'Остана да се наспиш';
        }
      }
      chasNextElem.innerHTML = nextState || 'Сладки сънища';
    }
    setInterval(checkTime, 1000);
```

Този JavaScript код дефинира функция, наречена checkTime(), която се използва за определяне на текущото състояние на графика, колко време остава до края или началото на часа и състоянието на следващото събитие. След това актуализира съдържанието на HTML елементите. Функцията се извиква всяка секунда с помощта на setInterval().

В началото функцията извлича препратки към три HTML елемента по техните идентификатори: "chasNow", "chasRemain" и "chasNext". Освен

това инициализира някои променливи: текущото време, низ nextState и булев флаг, наречен isEventFound.

След това кодът преминава през масива на графика, който вероятно съдържа обекти с информация за всяко събитие, включително начално и крайно време, както и състоянието. За всяко събитие той създава обекти startDateTime и endDateTime въз основа на началния и крайния час.

След това функцията проверява дали текущият час е в началния и крайния час на събитието. Ако е вярно, той изчислява оставащото време до края на събитието и актуализира съдържанието на елементите "chasNow" и "chasRemain". Той също така задава променливата nextState и флага isEventFound на true, след което излиза от цикъла.

Ако не бъде намерено събитие в текущото време, функцията изпълнява серия от условни оператори. Първо проверява дали текущият час е по-малък или равен на 13. Ако е вярно, той извлича състоянието и началния час на първото събитие, изчислява разликата във времето до началото на събитието и актуализира съдържанието на елементите "chasNow" и "chasRemain". След това задава променливата nextState на състоянието на третото събитие в графика.

Ако текущият час е по-голям от 13, съдържанието на елементите "chasNow" и "chasRemain" се актуализира с конкретен текст, което показва, че учебният ден е приключил.

И накрая, функцията актуализира съдържанието на елемента "chasNext" със стойността на променливата nextState или низ по подразбиране "Сладки сънища" (Sweet dreams), ако няма следващо състояние.

Функцията setInterval() в края на кода гарантира, че функцията checkTime() се извиква на всеки 1000 милисекунди или всяка секунда, за да поддържа показваната информация актуална.

4.2.5. ЗаМен

Тази страница представлява кратко описание на самия уебсайт и думите на автора, тоест Аз. Те са разделени по две заглавия, на които автора отговаря с добре оформен и четлив текст, за да може читателя да вникне в това, което искаме да кажем. Едно от заглавието е „Каква цел има този уебсайт?“, което всъщност е въпрос, с който описваме предназначението на уебсайта.

4.2.5.1. Оформление

4.2.5.2. Текст

Текста в тази страница се изобразява посредством инструмент, който трябваше да създадем и да персонализираме с помощта на библиотеката TypeIt.

```
<script src="https://unpkg.com/typeit@8.7.1/dist/index.umd.js"></script>
```

С горе показания код вкарахме библиотеката, от която се нуждаем в този случай. Библиотека има много възможности и приложения и ние можем да се възползваме от тях.

Както вече споменахме, ние разделихме основния текст на две заглавия. Първото заглавие се отнася за автора или създателя на уебсайта. С текст ние описахме кратка история, която поради създаването на уебсайта. Също така описахме идея и посланието, което този уебсайт предоставя пред други алтернативи.

```
<h1 class="head-text">За мен</h1>
      <p id="men" class="align-justify" style="margin-left: 10%;margin-right: 10%;font-family: Pattaya; padding-bottom: 500px;"></p>

<h1 class="head-text">Каква цел има този уебсайт?</h1>
      <p id="cel" class="align-justify" style="margin-left: 10%;margin-right: 10%;font-family: Pattaya;"></p>
```

Това е HTML код, който задава структурата на заглавието и текста. Състои се от две основни части: заглавие и параграф с идентификатор „men“, последван от друго заглавие и параграф с идентификатор „cel“.

Първото заглавие се дефинира с помощта на тага <h1> и има клас "head-text".

Първият абзац е дефиниран с помощта на тага <p> и има id "men". Има класа "align-justify" и някои вградени стилове, които задават полета, семейство шрифтове и подложки.

Второто заглавие се дефинира с помощта на тага <h1> и има клас "head-text". Текстът в заглавието е "Каква цел има този сайт?".

Вторият параграф се дефинира с помощта на тага <p> и има идентификатор "cel". Има класа "align-justify" и някои вградени стилове, които задават полетата и семейството шрифтове.

```
new TypeIt("#men", {
strings: ["текст."],
speed: 20,
waitUntilVisible: true,
cursor: false,
})
.go();
```

Този код използва библиотеката TypeIt, която се използва за създаване на ефект на писане върху HTML елемент с ID "men".

Обектът TypeIt се създава с конфигурационен обект, подаден като аргумент. Конфигурационният обект има следните свойства:

- Низове: масив от низове, през които ефектът на писане ще преминава. В този случай има само един низ.

- Скорост: Скоростта, с която ще се появи ефектът от въвеждане, в милисекунди на символ.
- WaitUntilVisible: Булева стойност, която определя дали ефектът от въвеждане ще започне веднага или ще изчака, докато HTML елементът стане видим в прозореца за изглед.
- Курсор: Булева стойност, която определя дали курсорът ще бъде показан по време на ефекта на писане.

След като обектът TypeIt е създаден, методът .go() се извиква, за да стартира незабавно ефекта на писане.

```
    new TypeIt("#cel", {
      strings: ["текст"],
      speed: 20,
      waitUntilVisible: true,
      cursor: false,
    })
      .move(null, { to: "END" })
      .break()
      .break()
      .type("Благодарим ви, че се отбихте и се надяваме да се насладите на
този празник на нашето гимназиално изживяване!")
      .go();
```

Този код използва библиотеката TypeIt, за да създаде ефект на писане в определен HTML елемент с id "cel". Съдържанието, което ще бъде въведено, е посочено в масива от низове, който съдържа единичен низ с дълъг абзац, описващ предназначението на платформата.

Опцията за скорост определя колко бързо се въвежда текстът, като стойност 20 показва, че всеки знак ще бъде въведен за 20

милисекунди. Опцията waitUntilVisible е зададена на true, което означава, че анимацията за въвеждане няма да започне, докато елементът с идентификатор "cel" не се вижда в прозореца за изглед.

Опцията за курсор е зададена на false, което означава, че няма да се показва курсор по време на анимацията за въвеждане.

След като първоначалният низ бъде въведен, екземплярът TypeIt след това се инструктира да премести курсора до края на въведения текст, като използва метода за преместване с опцията to, зададена на „END“. След това се вмъкват два нови реда, като се използва методът за прекъсване.

Накрая съобщението "Благодаря на всички. Чао!" се въвежда с помощта на метода type и методът go се извиква, за да започне анимацията за въвеждане.