Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „СИТ“

**Документация**

по дисциплината „Управление на софтуерни проекти”

на тема: „Система за търсене на автомобили“.

|  |  |
| --- | --- |
| Изготвили: Калоян Стоянов Петров, Мартин Димитров Владимиров, Атанас Димитров Димов. | Проверил: |
| Специалност: СИТ |  |
| Група:4б |  |
| Факултетен номер: 20621615, 20621678, 20621616. |  |

# 2023

Съдържание

[2023 1](#_Toc135179849)

[Техническо задание на проекта: 3](#_Toc135179850)

[Обосновка на избора за система за контрол на версиите: 3](#_Toc135179851)

[Причина за избора: 3](#_Toc135179852)

[Трудности при използването: 3](#_Toc135179853)

[Настройки, нужни за работа с него: 3](#_Toc135179854)

[Съпоставка на избрания софтуер с други: 6](#_Toc135179855)

[Обосновка на избора за система за управление на проекти: 6](#_Toc135179856)

[Причина за избора: 6](#_Toc135179857)

[Трудности при използването: 7](#_Toc135179858)

[Настройки, нужни за работата с него: 7](#_Toc135179859)

[Използвани функционалности: 11](#_Toc135179860)

[Обосновка и разяснения към избраната методология за разработка на софтуера: 19](#_Toc135179861)

[Причина за избора: 19](#_Toc135179862)

[Предимства на подхода: 19](#_Toc135179863)

[Недостатъци на подхода, открити при изпълнението на задачата: 20](#_Toc135179864)

[Съпоставка на подхода с други: 21](#_Toc135179865)

[Описание на стъпките по реализацията: 21](#_Toc135179866)

[Определяне на роли: 21](#_Toc135179867)

[Разделяне на задачи: 22](#_Toc135179868)

[Определяне на срокове за: проектиране, кодиране, документиране, интегриране, тестване: 22](#_Toc135179869)

[Избор на програмни средства за реализацията – програмен език, IDE: 23](#_Toc135179870)

# Техническо задание на проекта:

Система за търсене на автомобили. Системата трябва да предоставя следните възможности:

* въвеждане на информация за автомобил – напр. марка, модел, екстри и др.;
* търсене на автомобили, които отговарят на определени критерии (филтриране) – напр. марка, модел, екстри и др.

# Обосновка на избора за система за контрол на версиите:

## Причина за избора:

Когато става въпрос за системи за контрол на версиите няма особено богат избор на технологии. Избрахме Git защото е безплатна,open source и има изградена общност от потребители и интеграция, с почти всички среди за разработка(IDEs). При Git всеки разработчик работи върху свое работно копие на цялото хранилище. Позволява създаване и сравняване на различни версии на приложението и да връщането към предишни версии при нужда което гарантира, че кодът е актуален и работи коректно. Git позволява също така разработчиците да работят заедно в реално време върху същия код и да го споделят с помежду си. Малък е и бърз почти всички операции са фокусирани локално, което дава огромно предимство в скоростта на изпълнение на командите, за разлика например от централизираните системи, при които е необходима комуникация със сървъра. Представител е на клона системи за контрол на версиите, базирани на децентрализирано хранилище.

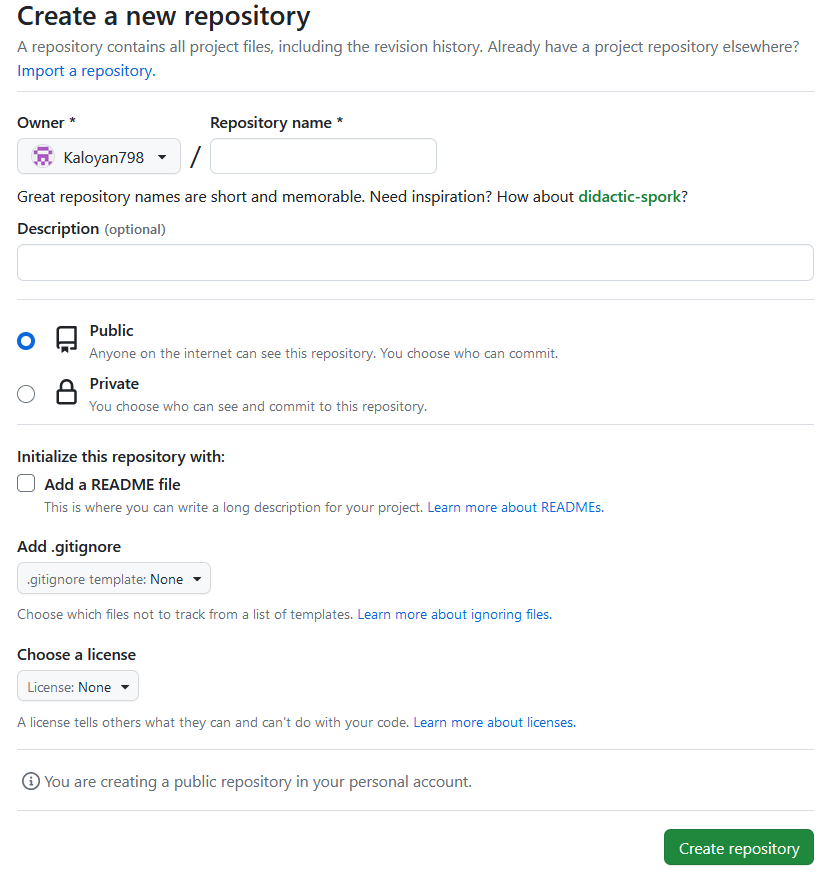
## Трудности при използването:

Една от най-големите трудности е стръмната крива на обучение. Git има уникален дизайн и работен процес, чието разбиране и овладяване може да отнеме известно време. Друг потенциален проблем е трудността при управлението на големи двоични файлове. Git е оптимизиран за текстови файлове, а големите двоични файлове като графики или видеоклипове могат да забавят системата и да затруднят работата с нея.

## Настройки, нужни за работа с него:

Изтеглете и инсталирайте git от <https://git-scm.com/downloads>.

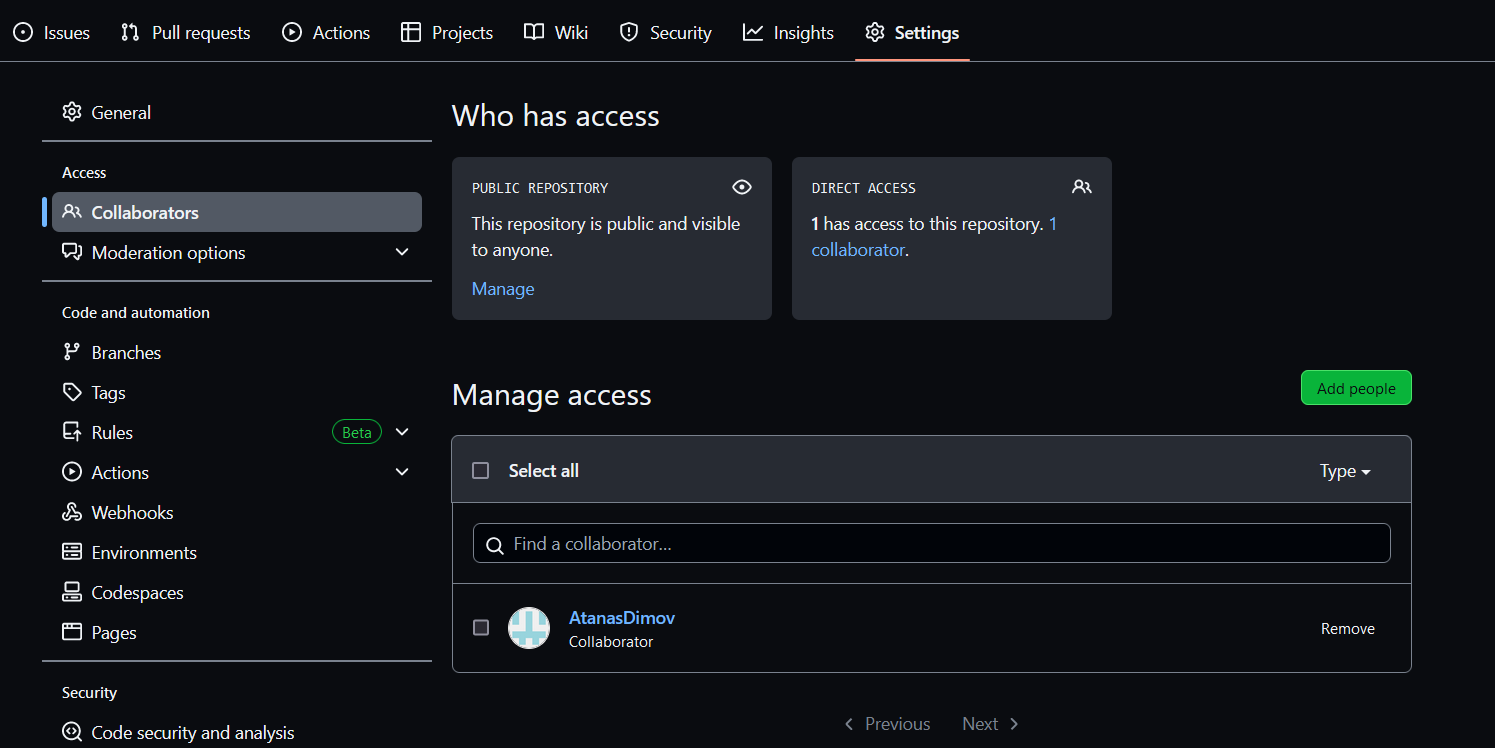
1. Създайте акаунт в github.com. Завършете регистрацията като избирате безплатен публичен план и потвърдете от e-mail.
2. Вписвате се в GitHub и след това е необходимо да стартирате проект, като изберете „**New repository**“(или +New Repository).

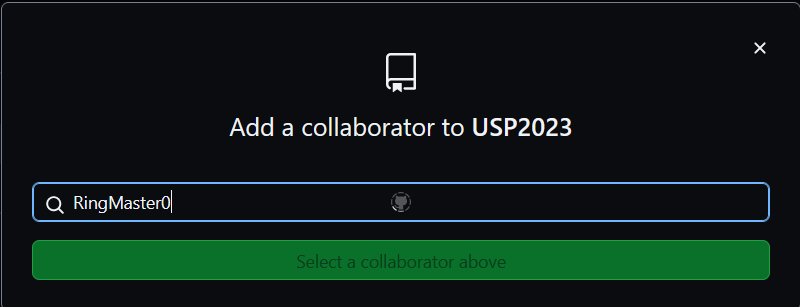


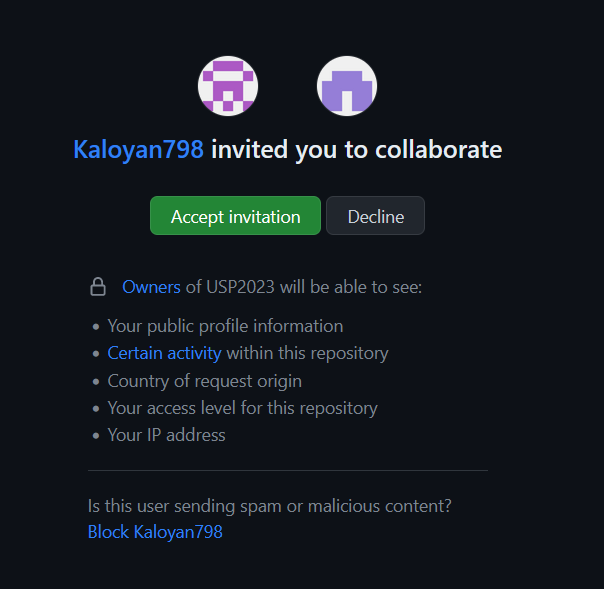
1. Във формата за създаване на ново хранилище имаме няколко задължителни полета:

* Име на хранилището (което следва да бъде уникално).
* Тип на хранилището (публично или частно).
* Описанието не е задължително.

1. Изберете"**Create repository**".:
2. След като сме се регистрирали и инициализирали хранилището е необходимо да се избере **Settings→Collaborators**, който искаме да добавим.



1. Необходимо е да се въведе потребителското име или e-mail-a на collaborator-a. След това се избира **→Add**
2. 
3. Collaborator-aе необходимо да си провери e-mail-a, да потвърди(**Accept Invitation**). По този начин се появява следния прозорец, който показва, че Collaborator-a вече е част от проекта.



Конфигуриране на Git във Visual Studio.

1. Отворете Visual Studio и отидете в менюто "**File**" (Файл).
2. Изберете "**Options**" (Опции) и след това "**Source Control**" (Управление на изходния код).
3. В полето "**Plug-in Selection**" (Избор на добавка) изберете "**Git**" от падащото меню.
4. След това, въведете име и имейл адрес, който ще се използва за вашите git комити.
5. За да конфигурирате глобалните настройки на git, отворете терминал в Visual Studio (например чрез менюто "**View**" (Изглед) -> "**Integrated Terminal**" (Интегриран терминал)) и изпълнете командите:

git config --global user.name "Вашето име"

git config --global user.email "Вашият имейл адрес"

1. След като настройките са конфигурирани, можете да започнете да използвате git във Visual Studio. За да отворите Git инструментите, щракнете върху менюто "**Team Explorer**" (Екипен център).
2. От тук можете да клонирате съществуващ репозитори или да създадете ново репозитори.
3. След като имате клониран репозитори, можете да работите със своите файлове и да ги комитвате промените в git.

Конфигуриране на Git във VS Code.

1. Отворете Visual Studio Code и отидете в менюто "**File**" (Файл).
2. Изберете "**Preferences**" (Предпочитания) и след това "Settings" (Настройки).
3. В полето за търсене въведете "**git.path**". Това ще покаже опциите за настройки за Git.
4. Изберете "**Edit in settings.json**" (Редактиране в settings.json), за да отворите файла с настройките.
5. В **settings.json** файла добавете следната настройка:

"git.path": "C:\\Program Files\\Git\\bin\\git.exe"

1. Затворете файла с настройки и рестартирайте Visual Studio Code.
2. Сега можете да отворите папка или проект, който съдържа Git репозитори в Visual Studio Code.
3. За да работите с Git, отворете терминала в Visual Studio Code, като изберете менюто "**Terminal**" (Терминал) и след това "**New Terminal**" (Нов терминал).
4. Използвайте Git командите в терминала, за да управлявате вашите Git репозитори, като например: **git clone, git add, git commit, git push** и други.

## Съпоставка на избрания софтуер с други:

Управлението на големи двоични файлове може да бъде трудно с Git и в такива случаи други системи за контрол на версиите като SVN или Mercurial може да са по-подходящи. В сравнение с други VCS като Mercurial, Git има по-бърза скорост на работа и позволява безопасна работа на и извън мрежата. Едно от най-големите предимства на Git е бранчването. За разлика от централизираните системи за контрол на версиите, бранчовете тук се създават, управляват и сливат много лесно.

# Обосновка на избора за система за управление на проекти:

## Причина за избора:

Trello

За нашия проект избрахме Trello. Това е софтуер, който помага за увеличаването на производителността при изпълнение наколективна работа. Trello предоставя дъски (boards), списъци (lists) и карти (cards), за да се организирати приоритизират проектите. Най-общо една дъска представлява един проект. Към нея може да сеизградят няколко на брой списъка – например TODO, IN PROCESS, DONE. В тези списъци се поставятзадачи – карти. След създаването на задача, към нея могат да се добавят коментари, да се прикачатфайлове, да се изготвят подсписъци за проверка (checklists), да се задват крайни срокове и др.

MS Project

Project 2010 представлява софтуерен инструмент за управление и автоматозиране на работатасвързана с управление на проекти. В него може да се въвежда информация за задачите по проекта,тяхното начало и продължителност на изпълнение, колко време ще отнеме изпълнението им и койтрябва да извърши работата.

## Трудности при използването:

Trello

Trello е полезен инструмент, но може да се срещнете с някои трудности при използването му. Изборът на правилната методология за използване на Trello може да бъде предизвикателство, особено ако имате голям проект. Безплатният план на Trello има някои ограничения, като например броя на командите и присъединените към тях хора, броя на картичките, които можете да създадете, и други. Освен това, ако използвате твърде много карти, списъци и етикети, Trello може да стане прекалено сложен и да загуби своята ефективност.

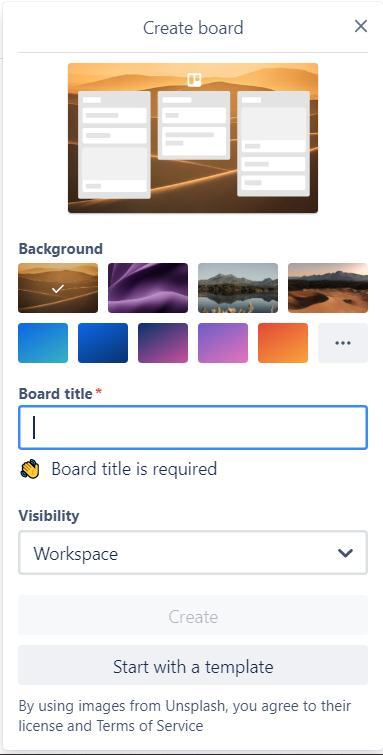
MS Project

При работата с MS Project, е възможно да се сблъскате с някои трудности. Първо, софтуерът може да бъде сложен за използване, особено ако сте нов потребител. Изисква се време и обучение, за да разберете всички функции и възможности на програмата. Второ, при големи и сложни проекти може да възникнат предизвикателства при управлението на множество задачи, ресурси и зависимости. Създаването и поддръжката на график, който отразява всички аспекти на проекта, може да изисква допълнителни усилия. Трето, управлението на ресурсите може да бъде предизвикателство, като балансиране на задачите и разпределение на ресурсите. Необходимо е да се осигури ефективно използване на ресурсите и избягване на прекъсвания. Освен това, ако работите в екип, съвместната работа и комуникацията могат да представляват предизвикателство. Необходимо е да се установят ясни процеси и комуникационни канали. Накрая, могат да се появят и технически проблеми, като съвместимост с различни версии на софтуера или възникване на грешки.

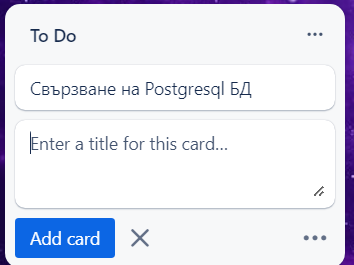
## Настройки, нужни за работата с него:

Trello

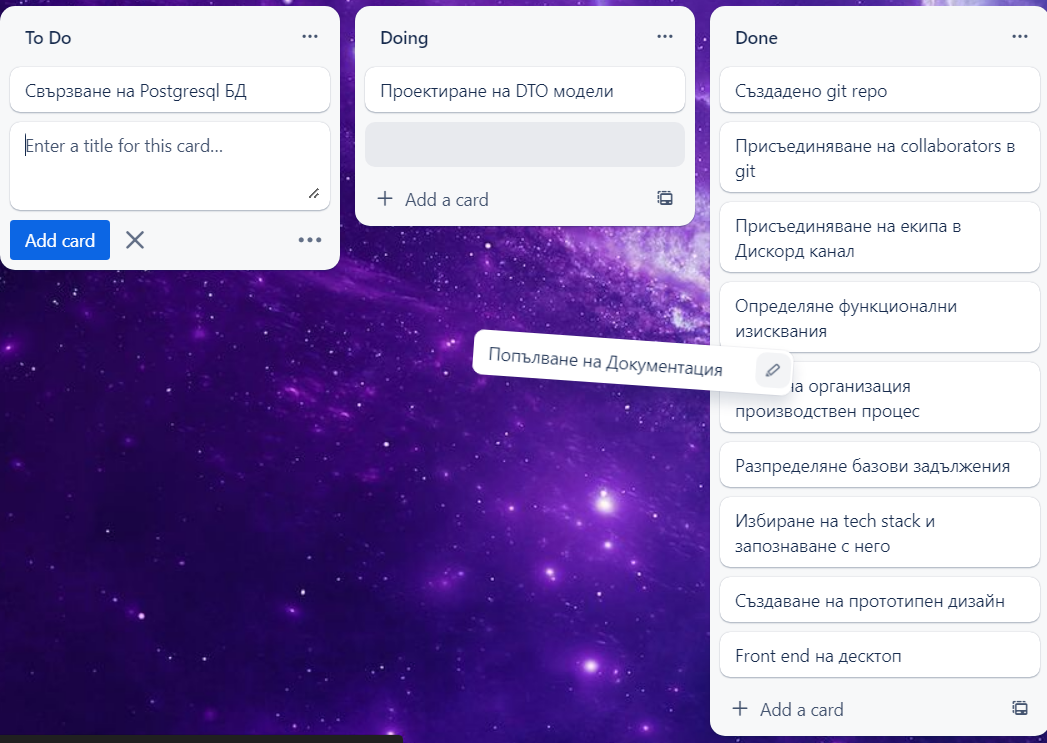
1. Създайте профил в Trello, като посетите https://www.trello.com на Trello и се регистрирате за безплатен акаунт.
2. Създайте дъски в Trello, като щракнете върху "Създаване на нова дъска". Дайте име на дъската и добавете списъци, които да включват картички.



1. Създайте картички, като щракнете върху бутона "Добавяне на карта". Дайте име на картичката и добавете описание, етикети, прикачени файлове и други.



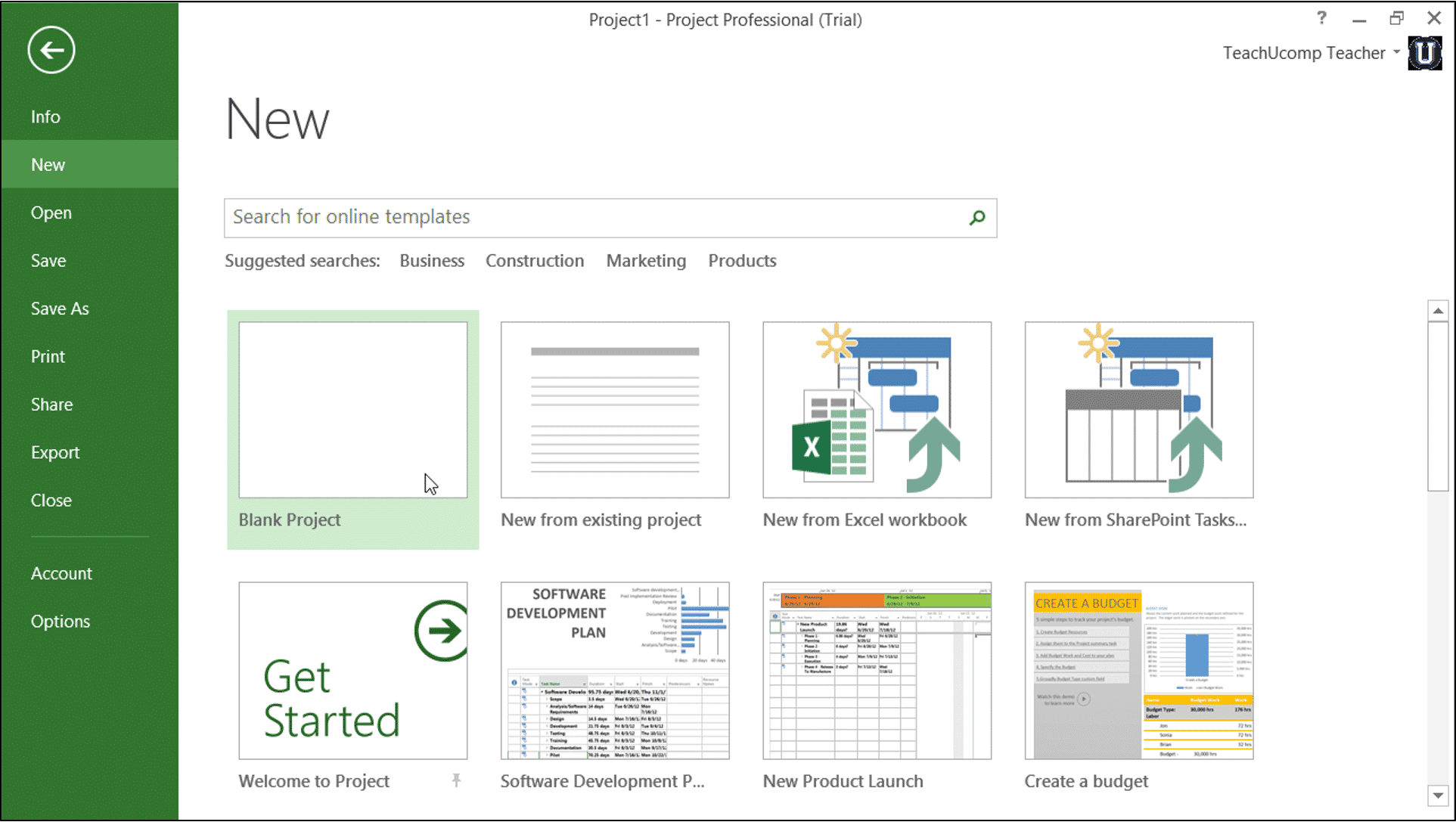
1. Преместете картички между списъците, като ги влачите и пускате с мишката. Можете да промените реда на списъците по същия начин.



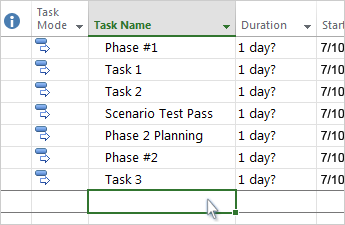
1. Създайте команди, като щракнете върху "Създаване на команда". Можете да добавите други потребители към командата и да им предоставите достъп до дъските.
2. Използвайте функциите на Trello, като добавите срокове, известия и други. Trello предлага много функции, които могат да бъдат използвани за управление на проекти.
3. Сътрудничете с други хора в дъските. Можете да приканите други потребители да се присъединят към дъските и да работите заедно по проектите.

MS Project

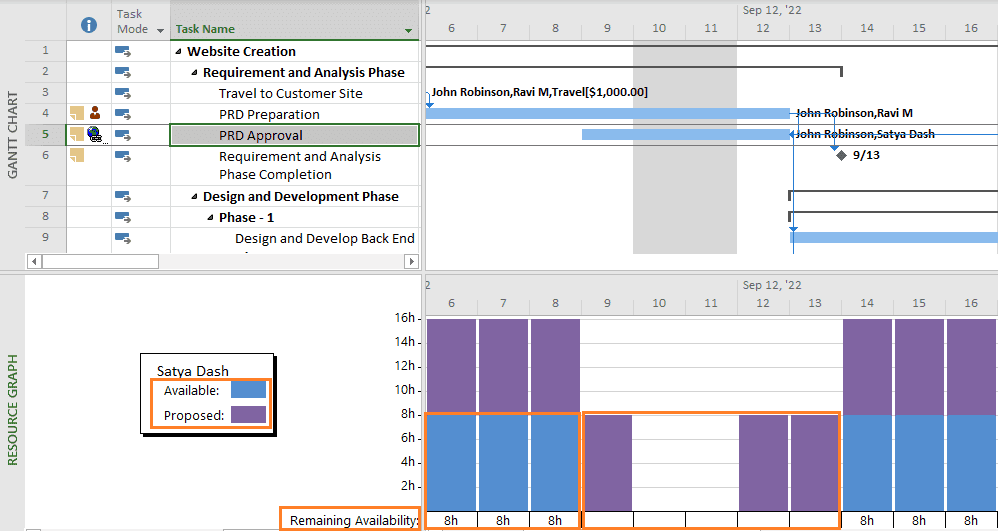
1. Създайте нов проект: Отворете MS Project и изберете опцията за създаване на нов проект. Можете да започнете от нулата или да използвате шаблон, ако имате предварително дефинирани настройки или структура на проекта.



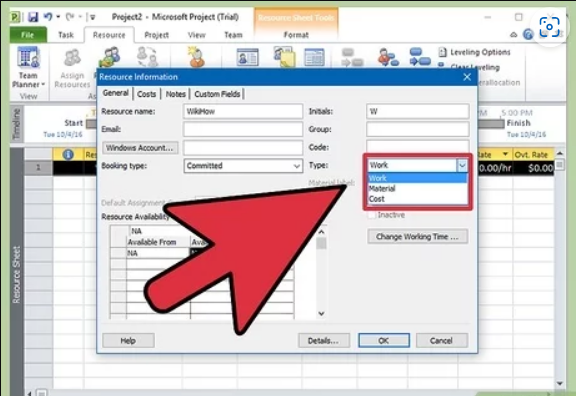
1. Дефинирайте задачите: Въведете задачите, които трябва да бъдат изпълнени в проекта. Определете дължината на задачите, зависимостите между тях и ресурсите, които са отговорни за тяхното изпълнение.



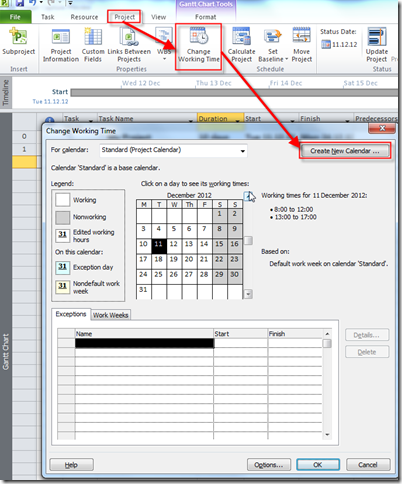
1. Настройте графика: Използвайте графика на MS Project, за да визуализирате последователността на задачите, продължителността им и зависимостите между тях. Можете да променяте графичния изглед, да филтрирате и групирате задачите, за да получите ясна представа за графика на проекта.



1. Добавете ресурси: Въведете ресурсите, които ще работят по задачите в проекта. Можете да определите техните работни часове, разпределение на работата и таксите им. MS Project ще ви помогне да управлявате наличността и използването на ресурсите във времето.



1. Настройте календарите: Дефинирайте календарите на проекта, включително работните и почивните дни, празниците и изключенията. Това ще ви позволи да проследявате и планирате работата в съответствие със специфичните календарни ограничения.



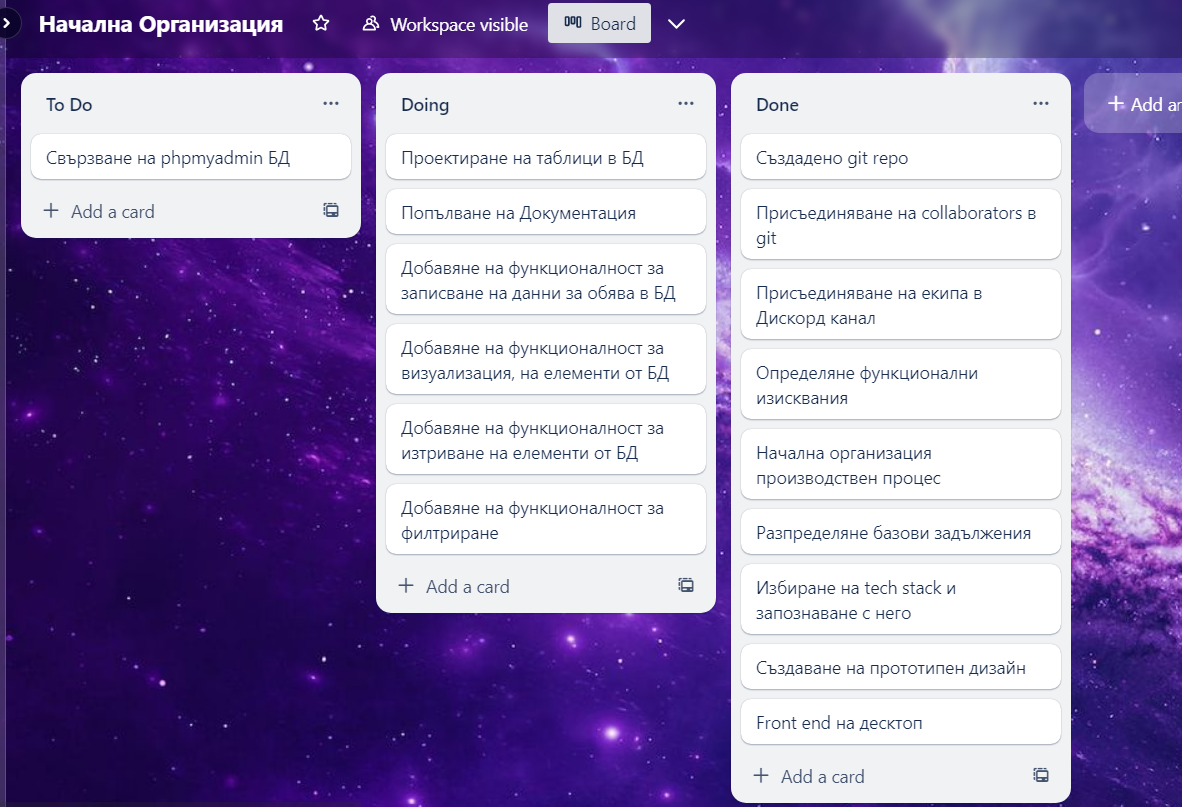
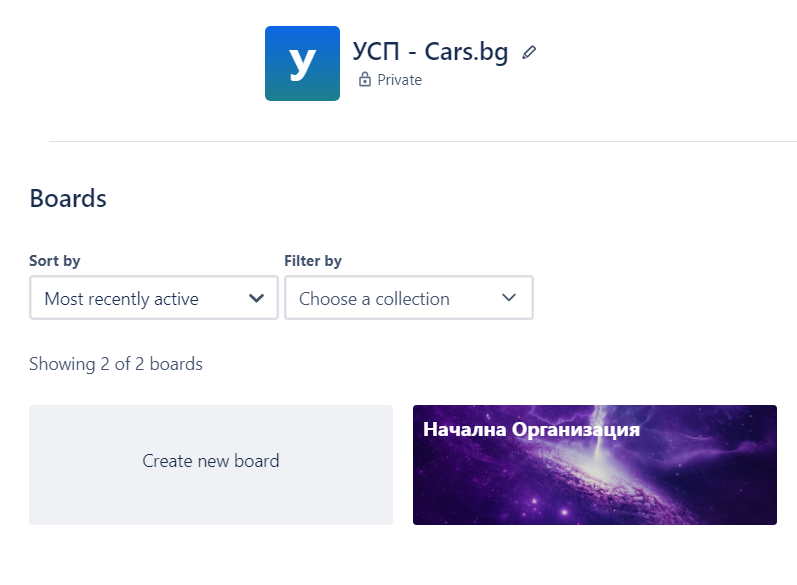
1. Прегледайте и оптимизирайте графика: Анализирайте графика и ресурсното използване на проекта. Извършете необходимите корекции и оптимизации, за да гарантирате, че проектът е реалистичен и изпълним.
2. Следете и актуализирайте напредъка: Редовно проследявайте и актуализирайте напредъка на проекта. Въведете фактически данни, като

## Използвани функционалности:

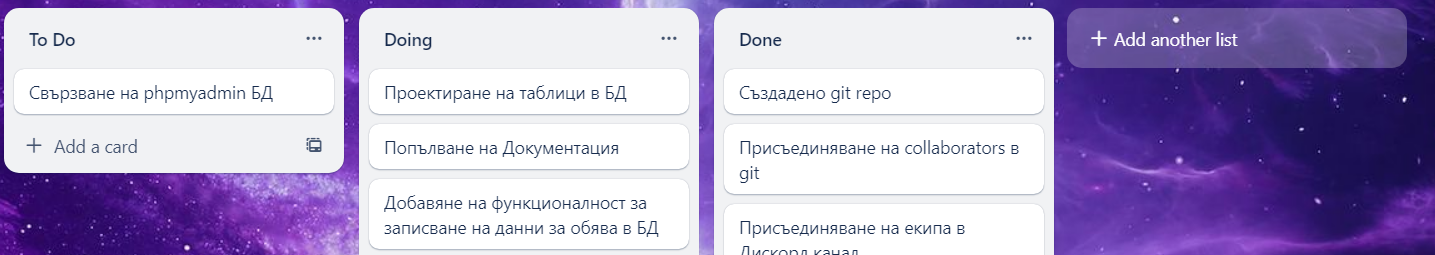
Trello

Ето някои от използваните функционалности на Trello:

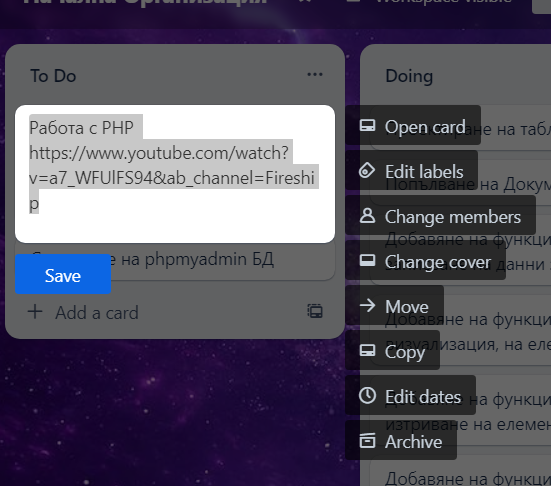
1. Дъски (Boards): Trello предоставя възможност за създаване на дъски, които представляват различни проекти, отдели или задачи. Дъските служат като контейнери за списъци и картички и позволяват структуриране и организация на информацията.



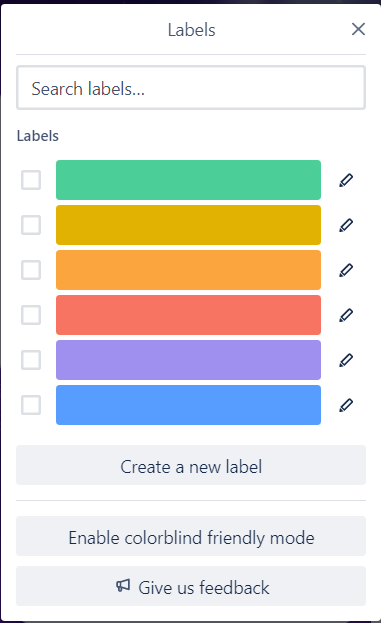
1. Списъци (Lists): Върху всяка дъска могат да бъдат добавени списъци, които представляват фазите, етапите или категориите на проекта. Например, списъци може да бъдат "To Do" (За изпълнение), "In Progress" (В процес) и "Done" (Изпълнено).



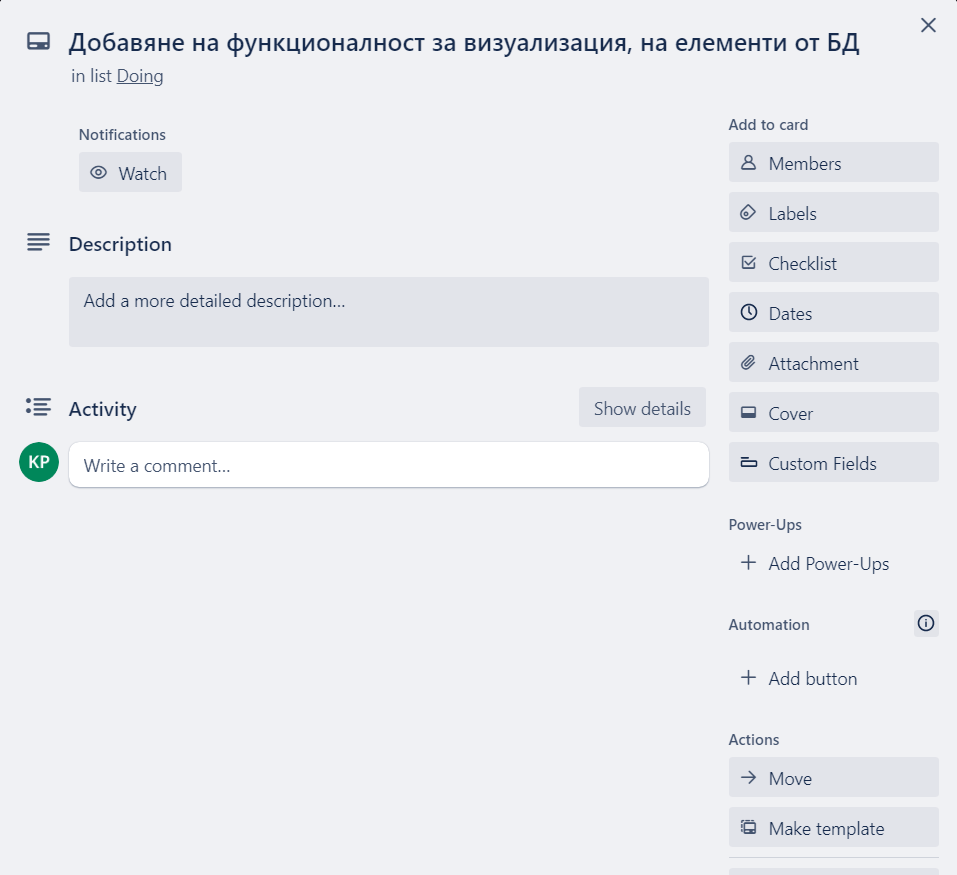
1. Картички (Cards): Върху всеки списък могат да бъдат добавени картички, които представляват отделните задачи, идеи, бележки или документи. Картичките могат да бъдат премествани между списъците, което отразява напредъка на задачите.



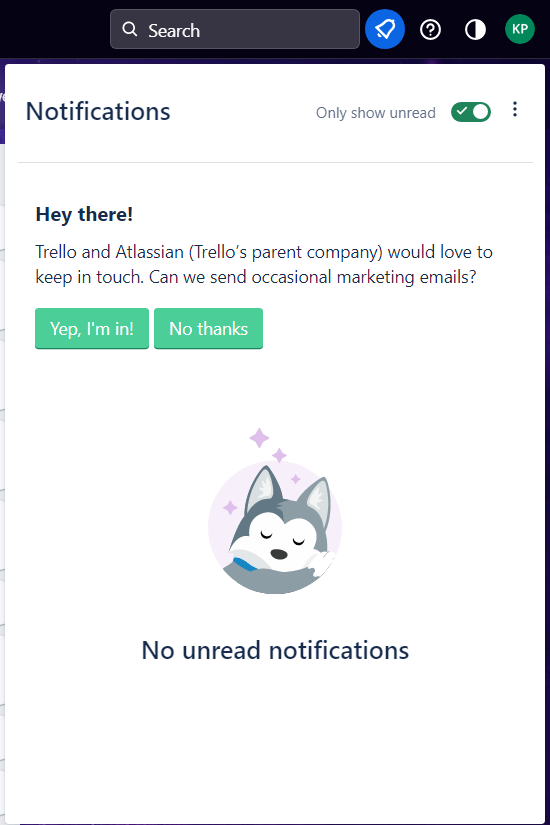
1. Етикети (Labels): Trello позволява добавянето на етикети към картичките, които могат да имат различни цветове и се използват за организация, маркиране на приоритети или идентифициране на специфични типове задачи.



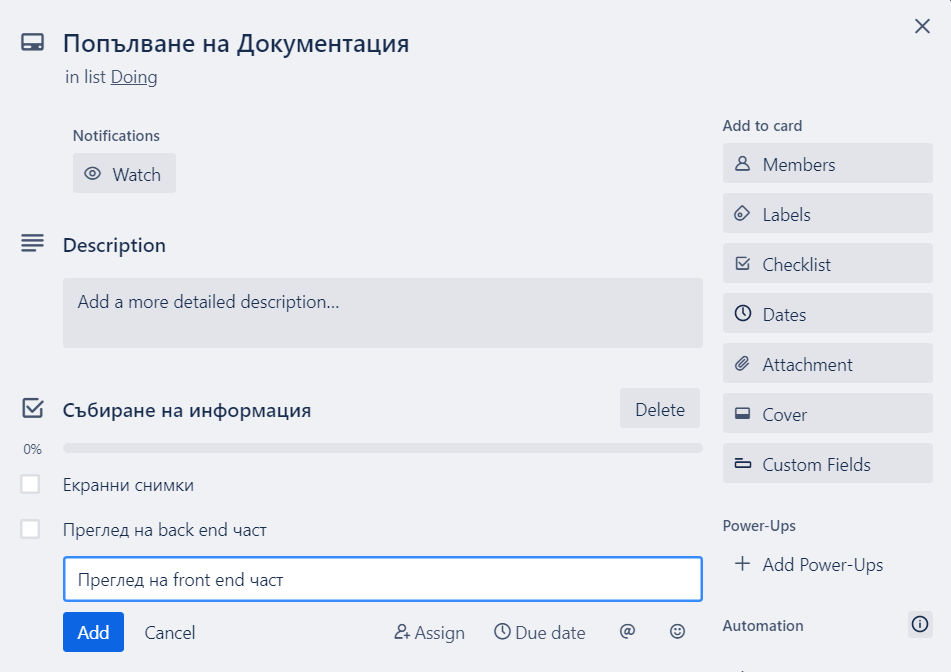
1. Коментари и прикрепени файлове: Върху всяка картичка могат да бъдат добавяни коментари и прикрепени файлове, което позволява екипът да комуникира и споделя информация за проекта.



1. Уведомления (Notifications): Trello предоставя функционалност за уведомяване, която дава възможност на потребителите да следят актуалната информация за промени, коментари или промени на картичките и дъските.

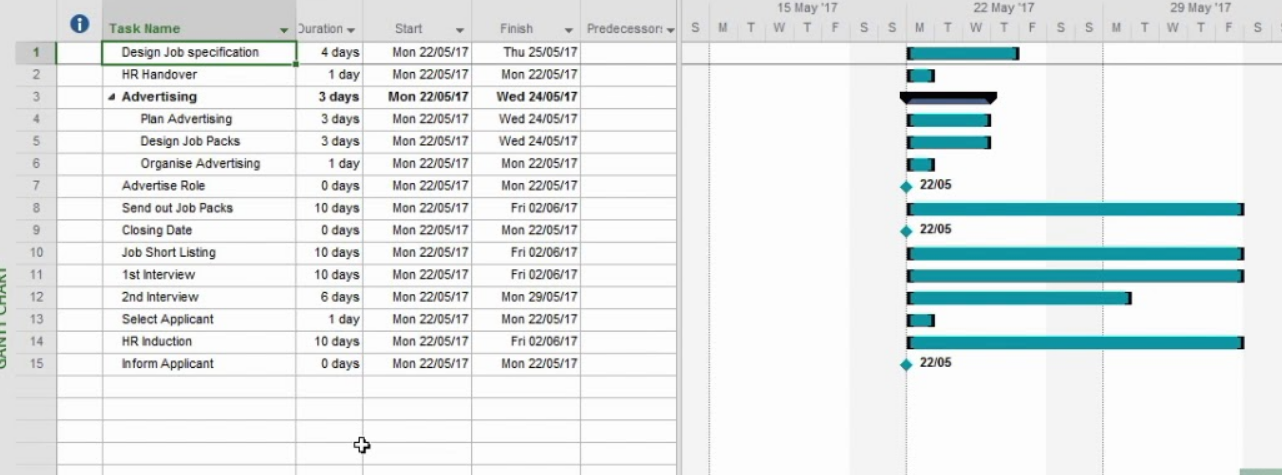


1. Чеклисти (Checklists): Може да се добавят чеклисти към картичките, което помага за структуриране на подзадачите или етапите на дадена задача и следене на напредъка.

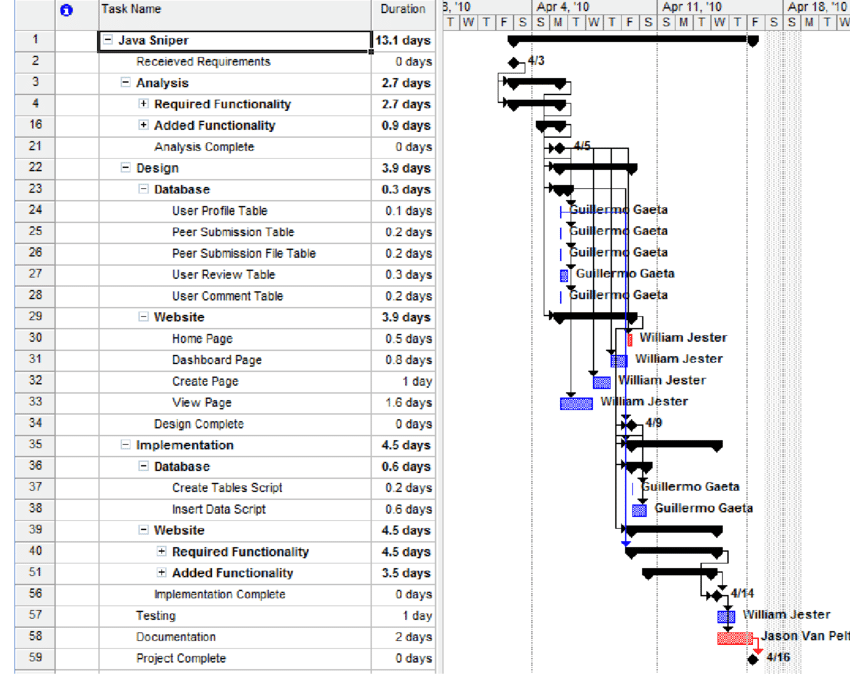


Ето някои от използваните функционалности на Microsoft Project:

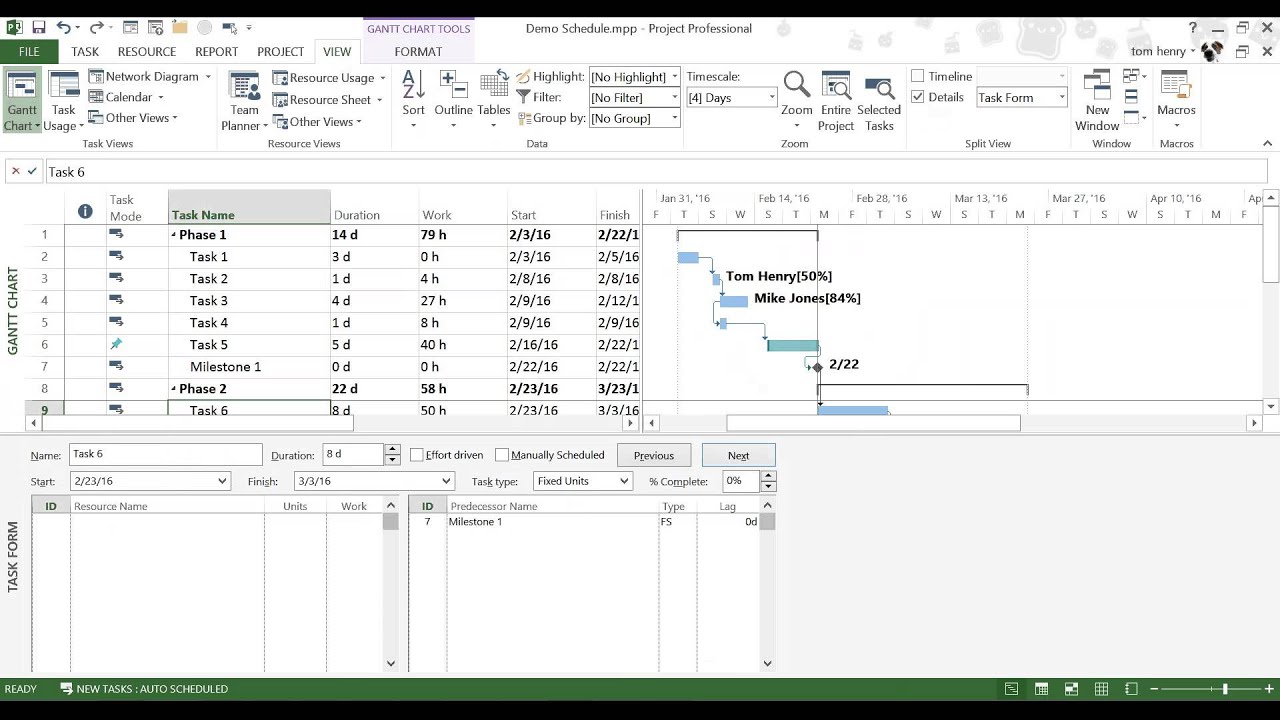
1. Създаване на задачи и йерархия на задачите: Microsoft Project позволява създаването на задачи и определянето на тяхната зависимост и йерархия. Можете да дефинирате продължителността на всяка задача, ресурсите, които ще работят по нея, и свързаните зависимости.



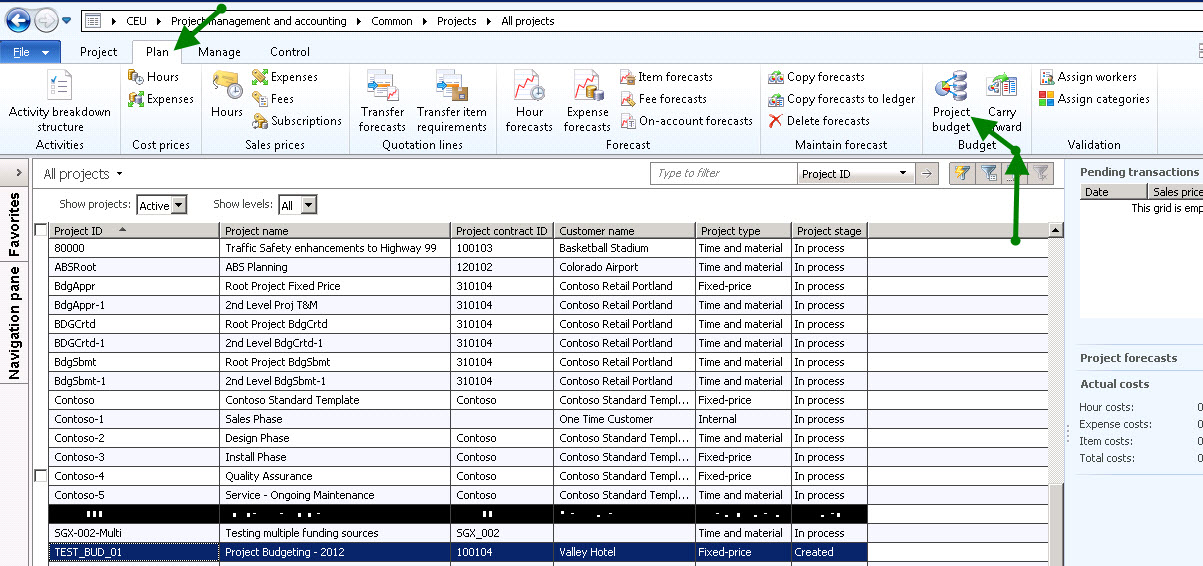
1. Гантови диаграми: Гантовите диаграми са инструмент, който визуализира задачите и графика на проекта. Microsoft Project предоставя възможност за създаване на Гантови диаграми, които представят времевата линия на проекта, задачите и зависимостите между тях.



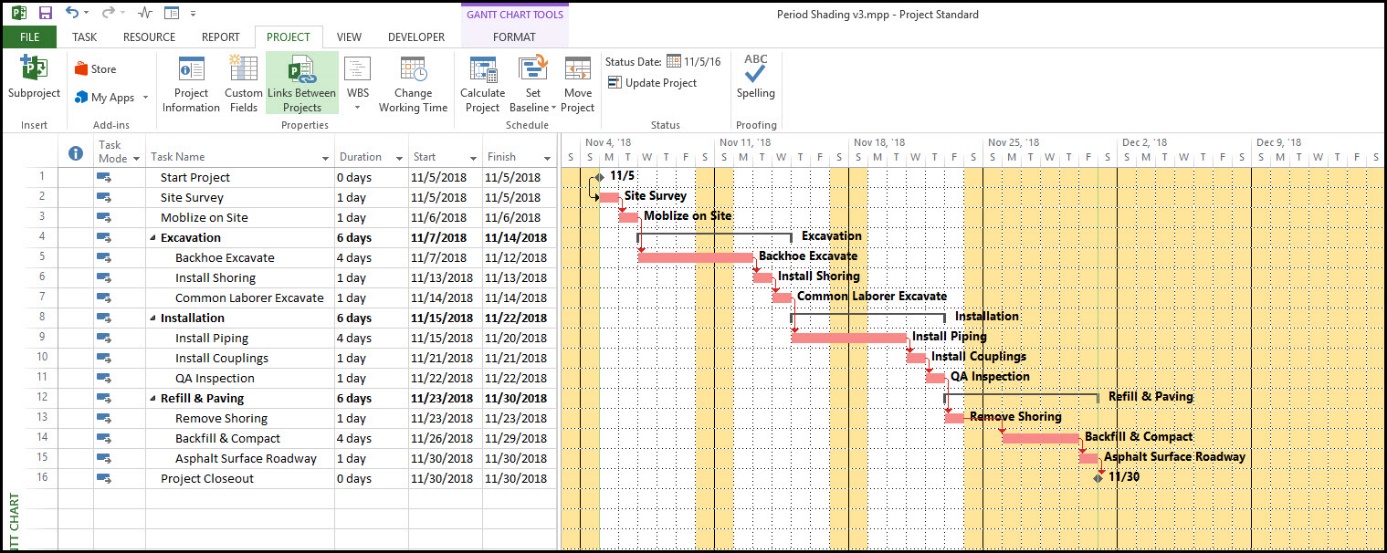
1. Ресурсно управление: Microsoft Project позволява управление на ресурсите, които участват в проекта. Можете да дефинирате различни типове ресурси (хора, оборудване, материали) и да определите количеството и разпределението им в рамките на проекта.



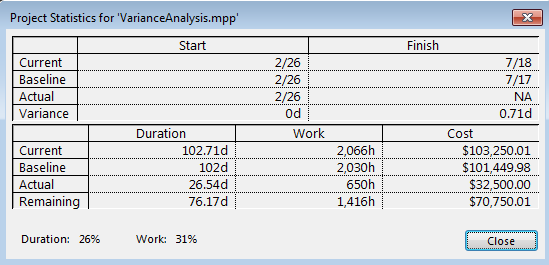
1. Бюджетиране и разходи: Microsoft Project предоставя възможност за създаване на бюджети на проекта и управление на разходите. Можете да дефинирате разходите за ресурсите, разходите за задачите и да следите общите разходи на проекта.



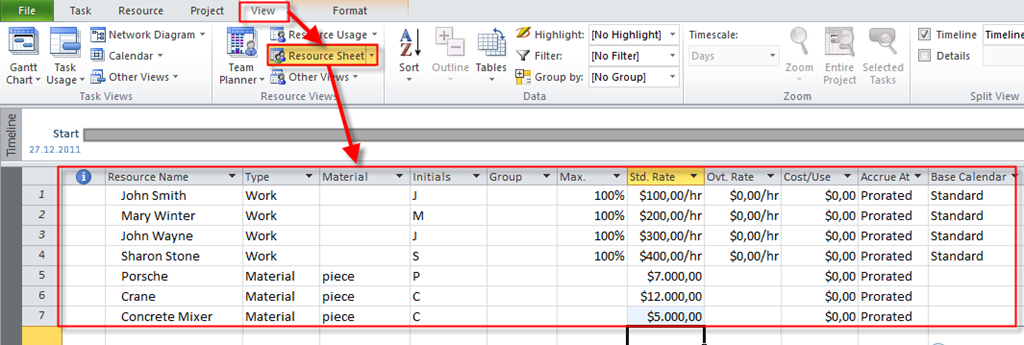
1. Управление на времеви графици: Microsoft Project ви позволява да планирате времеви графици за проекта, които отразяват времевата линия на задачите и зависимостите между тях. Можете да проследявате и актуализирате напредъка на задачите спрямо планираните срокове.



1. Отчети и анализи: Microsoft Project предлага голям брой вградени отчети и анализи за проекта. Можете да генерирате отчети за напредъка на проекта, разходите, ресурсите, критичните пътища и други важни параметри.



1. Сътрудничество и споделяне на данни: Microsoft Project позволява сътрудничество и споделяне на данни с други участници в проекта. Можете да споделяте графики, отчети и данни с колегите си и да работите заедно в реално време.



Тези функционалности на Microsoft Project помагат на управленски екипи и проектни мениджъри да планират, проследяват и контролират различни аспекти на проекта, включително задачите, ресурсите, времевите графици и разходите.

# Обосновка и разяснения към избраната методология за разработка на софтуера:

## Причина за избора:

Agile е методологията, която избрахме ние. Тя се фокусира върху гъвкавостта, прозрачността и непрекъснатото взаимодействие с клиента. Тя се основава на набор от принципи и практики, които подкрепят сътрудничеството между разработчиците и клиента и насърчават бързата итеративна разработка. Тази методология е изключително гъвкава и позволява на екипите да се приспособяват към променящите се изисквания и условия. Това е особено полезно в динамични и бързо променящи се среди, където изискванията могат да се променят по време на разработката.

1. Итеративен подход: Agile предлага итеративен подход, като разработката се разделя на малки итерации или "спринтове". Това позволява на екипа да получи бърза обратна връзка от клиента и да включи промени и корекции на ранен етап в процеса на разработка. Това води до по-добро управление на риска и възможност за приспособяване към нова информация.
2. Прозрачност и комуникация: Agile подкрепя прозрачността и отворената комуникация в екипа и с клиента. Разработчиците и клиентът работят заедно през целия процес на разработка, като споделят информация, идеи и проблеми. Това помага за изграждането на доверие и разбиране между всички участници и дава възможност за по-добро разбиране на нуждите и изискванията на клиента.
3. Бърза доставка на стойност: Agile методологията се фокусира върху бързата доставка на стойност за клиента. Чрез итеративния подход и постоянното взаимодействие с клиента, екипът може да се съсредоточи върху разработката на високоприоритетни функционалности и да г

и достави бързо. Това помага на клиента да вижда конкретни резултати и да получава полезни продукти в ранен етап на разработката.

1. Адаптивност и иновации: Agile подкрепя иновативността и експериментирането. Поради гъвкавия си подход, екипите имат възможността да тестват нови идеи, функционалности и подходи и да се адаптират бързо според резултатите. Това стимулира иновациите и постоянното развитие на продукта.

## Предимства на подхода:

Решението ни да се използва Scrum подход за разработка на софтуера е обосновано от следните причини:

1. Прозрачност и предсказуемост: Scrum подходът осигурява прозрачност на проекта чрез ежедневни срещи, редовни прегледи на спринтовете и инкрементална доставка на работещ софтуер. Това дава възможност на клиента и всички заинтересовани страни да имат ясно разбиране за напредъка и възможност за предсказуемост в отношение на времеви рамки и функционалности.
2. Адаптивност и гъвкавост: Scrum подходът е известен със своята гъвкавост и способност да се адаптира към променящите се изисквания и условия. Спринтовете, които са кратки времеви интервали, позволяват на екипа да преразгледа и промени плановете си на всеки спринт, като реагира бързо на променящите се обстоятелства и възможности.
3. Колаборация и сътрудничество: Scrum насърчава активното взаимодействие и сътрудничество в екипа. Ролите на Scrum Master, Product Owner и Development Team са ясно дефинирани и създават основа за ефективна комуникация и работа. Това помага за постигане на високо ниво на ангажираност и сътрудничество, което е от съществено значение за успешното разработване на софтуер.
4. Бърза доставка на стойност: Scrum подходът е насочен към бързата итеративна доставка на работещ софтуер. Това означава, че клиентът може да получи полезни и използваеми функционалности във всяка итерация, което позволява на тях и на екипа да видят резултати и да въведат корекции или промени по ранен етап.
5. Постоянна обратна връзка: Scrum подходът осигурява систематични срещи и прегледи, които дават възможност за постоянна обратна връзка от клиента и за извличане на уроци и опит от всяка итерация. Това помага на екипа да се развива, да избягва грешките и да подобрява процеса си непрекъснато.

Изборът ни на Scrum подход за разработка на софтуер е обоснован, защото той комбинира гъвкавостта и адаптивността на Agile методологията със структурата и управлението, което осигурява прозрачност и предсказуемост.

## Недостатъци на подхода, открити при изпълнението на задачата:

При разработването на онлайн приложение за продажба на автомобили, открихме някои от следните недостатъци на Scrum подхода:

1. Ограничено време за разработка: На екипа ни липсва опит в разработката на софтуер и може да се нуждаем от повече време за разработване на приложението. Scrum подходът обаче е ориентиран към краткосрочни итерации, което може да наложи ограничено време за разработка, което може да бъде трудно за нас.
2. Ограничени ресурси: Имаме ограничени ресурси за разработка на проекта, като например ограничен бюджет за инфраструктура и софтуерни инструменти. Този недостатък може да усложни прилагането на Scrum подхода, който се основава на гъвкавост и използване на висококачествени инструменти за разработка.
3. Отсъствие на опитен Scrum майстор: Нямаме опитен Scrum майстор в екипа си. Това водеше до затруднения в прилагането на Scrum подхода и можеше да се наложи да бъдат изпратени запитвания до асистентите или да се търси помощ от опитни колеги.
4. Необходимост от по-голямо участие на клиента: За да се гарантира, че сайта за продажба на автомобили отговаря на нуждите на клиентите, може да бъде необходимо по-голямо участие на клиента в процеса на разработка. Това може да се окаже трудно за нас, особено ако клиентът е зает с други задачи или не е наясно с изискванията си.

## Съпоставка на подхода с други:

Ето някои съпоставки между Scrum и други популярни подходи за разработка на софтуер:

1. Сравнение между Scrum и Waterfall (Каскадна методология):
   * Scrum: Scrum е итеративен и инкрементален подход, който се основава на гъвкавост, адаптивност и активно участие на клиента. Проектът се разделя на кратки времеви интервали (спринтове), като всяка итерация включва планиране, разработка, тестване и доставка на функционалност. Подходящ за проекти, които изискват гъвкавост и постоянна промяна на изискванията.
   * Waterfall: Waterfall е линеен подход, където разработката преминава през последователни фази като анализ, дизайн, разработка, тестване и доставка. Всяка фаза завършва преди започването на следващата. Подходящ за проекти с ясно дефинирани и стабилни изисквания.
2. Сравнение между Scrum и Kanban:
   * Scrum: Scrum има ясно дефинирани роли (Scrum Master, Product Owner, Development Team) и предписани събития (планиране на спринт, дейли стендъп, преглед на спринт, ретроспектива). Фокусът е върху итеративността и инкременталността в разработката на софтуера.
   * Kanban: Kanban е визуална методология за управление на работата. Тя се базира на плъзгащи се колони, където задачите се движат от една колона в друга, отразявайки процеса на разработка. Подходящ за екипи, които се нуждаят от гъвкавост и непрекъсната видимост на работата.
3. Сравнение между Scrum и Lean:
   * Scrum: Scrum е ориентиран към итерации, в които екипът работи върху определени функционалности и възвръща стойност за клиента. Използва принципи като самоорганизация, взаимодействие и непрекъснато учене.

* Lean: Lean е методология за оптимизиране на процеси и премахване на излишни стъпки и ресурси. Съсредоточава се върху идентифициране и премахване на загуби и ефективност в разработката на софтуера.

Всеки от тези подходи има своите предимства и недостатъци и изборът на подход трябва да бъде базиран на нуждите на проекта, екипа и клиента. Scrum е популярен, защото предоставя гъвкавост, сътрудничество и активно участие на клиента, което може да бъде полезно при разработка на проекта.

# Описание на стъпките по реализацията:

## Определяне на роли:

Определянето на роли за back-end, front-end, документация и тестване е важна част от успешното разработване на онлайн приложение за продажба на автомобили. Ето някои от основните роли и отговорности, свързани с всяка област:

1. Back-end разработчик – Атанас.
2. Front-end разработчик – Мартин.
3. Роля за документаране и Тестер – Калоян.

## Разделяне на задачи:

1. Back-end разработчик - Атанас:
   * Отговаря за разработването и поддръжката на серверната част на приложението.
   * Работи с програмни езици и технологии като PHP и MySQL.
   * Изгражда и поддържа базата данни, извършва бизнес логиката и обработва заявките от клиентите.
   * Оптимизира производителността, сигурността и скалируемостта на системата.
2. Front-end разработчик - Мартин:
   * Отговаря за разработването на потребителския интерфейс и клиентската страна на приложението.
   * Работи с HTML, CSS и съответните библиотеки като Jquery и фреймуърк като SASS.
   * Създава дизайн, интерактивност и визуални ефекти, които осигуряват добра потребителска изживяване.
   * Уверява се, че приложението е отзивчиво и оптимизирано за различни устройства и браузъри.
3. Роля за документаране и Тестер - Калоян:
   * Отговаря за създаването и актуализирането на техническа документация и ръководства за приложението.
   * Документира архитектурата, функционалностите, API-тата, интеграциите и други технически аспекти на приложението.
   * Помага за осигуряване на ясност и разбираемост на разработчиците, които ще работят върху проекта, както и за лесното мащабиране и поддръжка на приложението.
   * Отговаря за планирането, изпълнението и документирането на тестови случаи за приложението.

* Открива и докладва за грешки и проблеми, свързани с функционалността и изпълнението на приложението.

## Определяне на срокове за: проектиране, кодиране, документиране, интегриране, тестване:

1. **Проектиране:** 2 седмици
2. **Кодиране:** 4 седмици
3. **Документиране:** 1 седмица
4. **Интегриране:** 1 седмица
5. **Тестване:** 1 седмица

## Избор на програмни средства за реализацията – програмен език, IDE:

Изборът на програмни средства за реализацията на онлайн приложение за продажба на автомобили е важна стъпка за постигане на успеха на проекта. Ето някои разяснения за избора на конкретни програмни езици, интегрирани среди за разработка (IDE) и технологии за front-end и back-end разработка:

Front-end:

1. HTML (HyperText Markup Language): Използва се за създаване на структура и семантика на уеб страниците.
2. CSS (Cascading Style Sheets): Използва се за стилизиране и визуално оформление на уеб страници.
3. jQuery: JavaScript библиотека, която предоставя множество функции и методи за улесняване на манипулациите с DOM елементите и обработка на събития.
4. Visual Studio Code (VS Code): Лека и мощна IDE, която предоставя инструменти за разработка на уеб приложения, включително поддръжка на HTML, CSS и JavaScript.

Back-end:

1. PHP: Скриптов език за уеб разработка, който се използва за създаване на динамични уеб страници и уеб приложения.
2. MySQL: Релационна база данни, която се използва за съхранение и управление на данните на приложението.
3. Visual Studio Code (VS Code): Предоставя удобна среда за разработка на PHP и интеграция с MySQL бази данни.

Изборът на тези програмни езици, технологии и среди за разработка може да се обоснове с тяхната популярност, поддръжка и подходящост за целите на проекта. HTML, CSS и jQuery са стандартни технологии за front-end разработка, които осигуряват добро визуално представяне и интерактивност на уеб страници. PHP и MySQL са широко използвани за back-end разработка, предоставяйки солидна функционалност и лесна интеграция с бази данни.

VS Code е популярна и мощна IDE със забележителни възможности за разработка на уеб приложения, които отговарят на изискванията на екипа от студенти.

* + стъпки, зависещи от конкретния подход за разработка;
* представяне на междинните timeline-и на проекта и на изходния timeline на проекта - според избрания подход за разработка би трябвало да направят акценти на специфичните за подхода на разработка моменти – ограничение на броя задачи; последователностите в състоянията на задача и др. Тук би трябвало да се разгледат промените в изискванията на възложителя.
* Изводи:
* Отчети, генерирани от софтуера за управление на проекти, и анализ на отчетите;
* Слаби моменти/трудности в изпълнението на задачата.