Univerzitet u Beogradu

Elektrotehnički fakultet



GITHUB

|  |  |  |
| --- | --- | --- |
| Mentor: |  | Student: |
| Dr Ana Gavrovska,  Docent |  | Aleksa Nikolić  2018/0306 |

Beograd, februar 2024

Sadržaj

[Uvod 3](#_Toc157792867)

[Git 4](#_Toc157792868)

[GitHub 5](#_Toc157792869)

[Postavka i izrada projekta 7](#_Toc157792870)

[Važne Git komande i njihov izgled 7](#_Toc157792871)

[GitHub deo projekta 9](#_Toc157792872)

[Zaključak 15](#_Toc157792873)

[Literatura 16](#_Toc157792874)

# 

# 

# Uvod

U današnjem globalističkom društvu, nije retko videti globalne saradnje različitih timova u praktično svim sferama rada. Usklađivanje remote članova projekta svakako je jedna od najvećih prepreka takvim vrstama saradnji, ali uz moćne alate kao što su Git i GitHub, to postaje jedna vrlo premostiva prepreka.

GitHub je web-based platforma dizajnirana za kontrolu verzije, kolaborativni razvoj softvera i razmenu koda. Omogućava efikasnu saradnju različitih timova time što pruža moćan alat za praćenje promena u kodu (ili generalno na projektu), organizaciji projekta i omogućavanju brze, lake i efikasne komunikacije između različitih članova tima ili čak timova.

Korisnici mogu da naprave repozitorijume gde mogu da skladište i organizuju datoteke vezane za projekat, usklađuju svoje doprinose sa drugim članovima pomoću branches (grana) i pull zahteva kao i mnogih drugih funkcionalnosti koji ovaj alat pruža.

GitHub je postao praktično centar saradnje za open-source razvojna okruženja, i svedoci smo sve većoj njegovoj upotrebi u svim mogućim projektima, gde on predstavlja nezamenjivu okosnicu saradnje. On promoviše transparentnost, kolaboraciju i uopšte inovaciju u globalnom društvu.

Uzevši sve ovo u obzir, nije začuđujuća činjenica da preko 90% softverskih projekata širom sveta koristi Git.

Cilj ovog rada je istaći korisne aspekte Git-a i GitHuba, pokazavši šta ih to tačno čini nezamenjivim članom svakog grupnog projekta, uz prikaz jednog primera kako GitHub može pomoći za jedan video projekat. Takođe je potrebno istaći i korisnost GitHub-a kao jednog globalnog centra gde osim grupnog rada možemo koristiti open source projekte drugih timova, što je takođe demonstrirano u prethodno navedenom primeru.

# Git

Git je distributed version control system (DVCS) koji se masovno koristi za praćenje promena koda pri razvoju softvera. Prva verzija napravljena je 2005 godine I od tada postao je praktićno standard za kontrolu verzije u industriji.

Najvažnije karakteristike i prednosti Django programskog jezika su:

1. **Kontrola verzije**: Git dozvoljava različitim članovima da prate promene projekta tokom vremena. Svaka promena se arhivira što omogućava prost pregled projekta od strane svih članova. On omogućava da više saradnika mogu istovremeno raditi na projektu a da nema međusobne smetnje.
2. **Distribuirani sistem:** Git je distribuirani sistem kontrole verzije, što znači da svaki korisnik ima kompletnu kopiju čitavog repozitorijuma, uključujući i čitav istorijat. Ovo omogućava nezavistan i čak offline način rada.
3. **Granjanje i spajanje:** Grananje je jedan od osnovnih mehanizama gita. Ono dozvoljava članovima tima da podele razvoj projekta u paralelne tokove. Zbog toga je lako raditi na novim funkcionalnostima a da pri tome ne remetimo glavni tok. Spajanje zapravo predstavlja obrnut process gde možemo spojiti različite tokove i time omogućiti saradnju različitih članova.
4. **Istorija i log:** Git beleži kompletnu istoriju promena, uključujući KO je napravio promenu, KADA, i ZAŠTO (putem commit poruka). Ovo pomaže u razumevanju kako i zašto se kod menjao tokom vremena.
5. **Saradnja:** Git olakšava saradnju tako što pruža mehanizme koji rešavaju problem koji mogu nastati kada više različitih članova želi da modifikuje isti fajl u isto vreme.
6. **Pregled koda:** Git pomaže u pregledu koda tako što olakšava praćenje promena u kodu. Članovi tima mogu da pregledaju promene, pruže feedback, i međusobno prodiskutuju promene pre nego što ih spoje u glavni deo projekta.
7. **Backup:** Pošto svaki clan tima ima kompletnu kopiju repozitorijuma, rizik gubitka podataka je minimalizovan. Ako bilo koji clan izgubi lokalnu kopiju, lako mogu povratiti igubljeni sadržaj sa zajedničkog repozitorijuma.
8. **Open source:** Git je open source, i poseduje ogromu zajednicu koja aktivno održava i doprinosi njegovom razvoju.
9. **Brzina i efikasnost:** Git je dizanjiran sa efikasnošću na umu. Operacije poput komitovanja promena, granjanja i spajanja su vrlo brze čak i za velike projekte.

# GitHub

Git i Github su vrlo usko povezani, gde je GitHub u suštini online platform koja koristi funkcionalnosti Gita, te sve što je rečeno za Git u suštini važi i za GitHub. Prednosti koje pruža GitHub su mnoge, ali i neophodne kako bi se uistinu realizovalo ono što Git pruža kao mogućnost. Važno je napomenuti kako GitHub nije jedina online platform bazirana na Git-u, već najpopularnija, te je praktično postao standard koji je u neku ruku sinoniman sa Git-om.

Bitno je razumeti kako se GitHub razlikuje od Git-a, ali i neke pojedine funkcionalnosti koje možda nemaju u Git-u:

1. **Centralizovano hostovanje repozitorijuma:**

Za razliku od Git-a, koji dozvoljava kontrolu verzije lokalno, GitHub pruža centralizovanu platformu koja hostuje repozitorijume. Developeri mogu vrlo lako napraviti repozitorijum koji će da skladišti njihov projekat u cloud-u.

1. **Korisnički interfejs**

Git, kao što će malo kasnije u ovom radu biti prikazan, koristi se preko komandne linije. Zahteva preciznu sintaksu koja mora da se nauči kako bi se baratalo Git-om. GitHub nudi korisnički interfejs koji uprošćava rukovođenje repozitorijumom i time pruža mogućnost korišćenja povoljnosti korisnicima koji možda ne žele da koriste kontrolnu liniju.

1. **Saradnja i forking**

GitHub omogućava saradnju i paralelni razvoj projekata time što omogućava takozvano forkovanje repozitorijuma. Forkovanje zapravo pravi kopiju originalnog repozitorijuma sa svim njegovim granama, omogućavajući slobodnu manipulaciju celokupnim projektom od strane skroz drugih članova ili timovima a da time ne remete originalni projekat. Ukoliko žele mogu da predlože svoje revizije originalnom projektu putem pull zahteva. Ovo omogućava znatno brz razvoj open source projekata.

1. **Pull zahtevi**

Kao što je prethodno pomenuto, Git pruža mogućnost korišćenja granjanja i spajanja, ali fali mu centralizovani proces predlaganja promena. GitHub uvodi upravo taj koncept koji se naziva Pull zahtev, koji je standardizovan i vizuelan način predlaganja promene i diskutovanja prednosti date promene. Oni pružaju prost način da timovi usaglase stavove i želje za datu izmenu pre nego što odluče da li će spojiti promene u glavni kod.

1. **Praćenje problema**

Git nema ugrađenu funkcionalnost praćenja mogućih bug-ova, problema, zahteva za novim funkcionalnostima itd. GitHub uvodi jedan robusan sistem koji dozvoljava korisnicima da prijave greške, bug-ove, da traže nove funkcionalnosti, diskutuju poboljšanja itd. Svaki od ovih zahteva može biti dodeljen konkretnom članu, labelisan, prioritizovan itd. Što pruža jedan vrlo praktičan i organizovan pristup upravi projektnih zadataka.

1. **Wiki i dokumentacija**

GitHub ima ugrađenu podršku za wiki, što omogućava timovima da prave i održavaju dookumentaciju projekta. Vrlo je korisno kako bi pružilo okosnicu toga kako je projekat nastao, od prvih koraka pa do realizacije.

1. **GitHub akcije**

GitHub akcije su CI/CD (Kontinualna integracija/Kontiunalno korišćenje) servis koji omogućava automatizovanje workflow-a, uključujući pravljenje, testiranje i primenu koda direktno u repozitorijum.

1. **Zajednica i društveno kodiranje**

GitHub podstiče društveno kodiranje time što olakšava developerima da otrkiju, prate i doprinose open source projektima. Javni repozitorijumi na GitHubu su pristupačni svim članovima. Samim tim, moguće je deljenje i kolaboracija na projektima.

Rezimirano, GitHub nadograđuje mogućnosti Gita time što pruža jedan centralizovan, online pristup sa dodatnim mogućnostima za saradnju poput pull zahteva, forkovanja, praćenja problema. Postao je centar za razvoj softvera, pogotovu za open source proojoekte i igra ključnu ulogu u omogućavanju kolaboracije i komunikacije developera, a da pri tome omogućava pristup i široj klijenteli jer ga mogu koristiti članovi koji nisu skloni korišćenju kontrolne linije.

# Postavka i izrada projekta

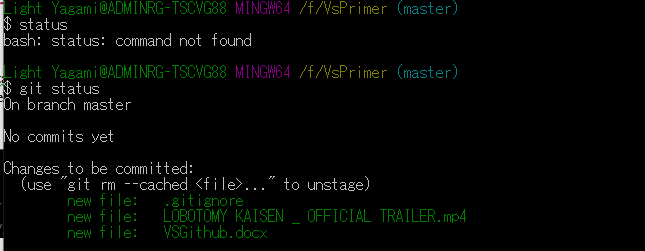
Cilj ovog projekta je, osim teoretski izneti i prikazati neke od funkcionalnosti Git-a i GitHub-a, takođe u jednom relanom primeru pokazati kako ovi alati mogu biti od koristi jednom video projektu. Za video projekat korišćen je GitBash, kao i Anaconda,Pytorch i naravno GitHub. Video obrada orađena je u DaVinci resolve video editing okruženju.

# Važne Git komande i njihov izgled

Prva i ključna stvar je inicijalizacija git repozitorijuma. To se postiže tako što se u komandnom prozoru prvo prebaci direktorijum u željeni folder, a onda rečju init inicijalizuje i arhivira folder u git repozitorijum. 

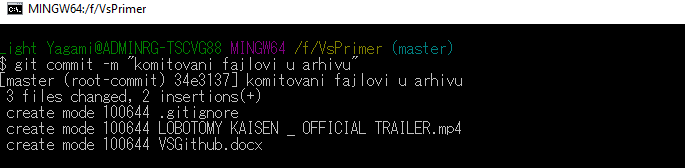
Slika 1. Command window incijalizacije git-a

Nakon toga neophodno je da Git prati fajlove datog foldera. To možemo postići komandom add



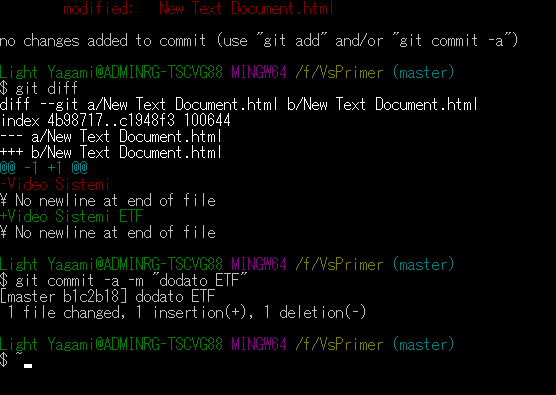
Slika 2. Command window za status git fajlova

Git sada prati naše fajlove ali nije ih još uvek komitovo. U suštini, fajlovi još nisu upisani u Gitovu istorijsku arhivu. To možemo postići komandom git commit.



Slika 3. Command window commit-a u gitu

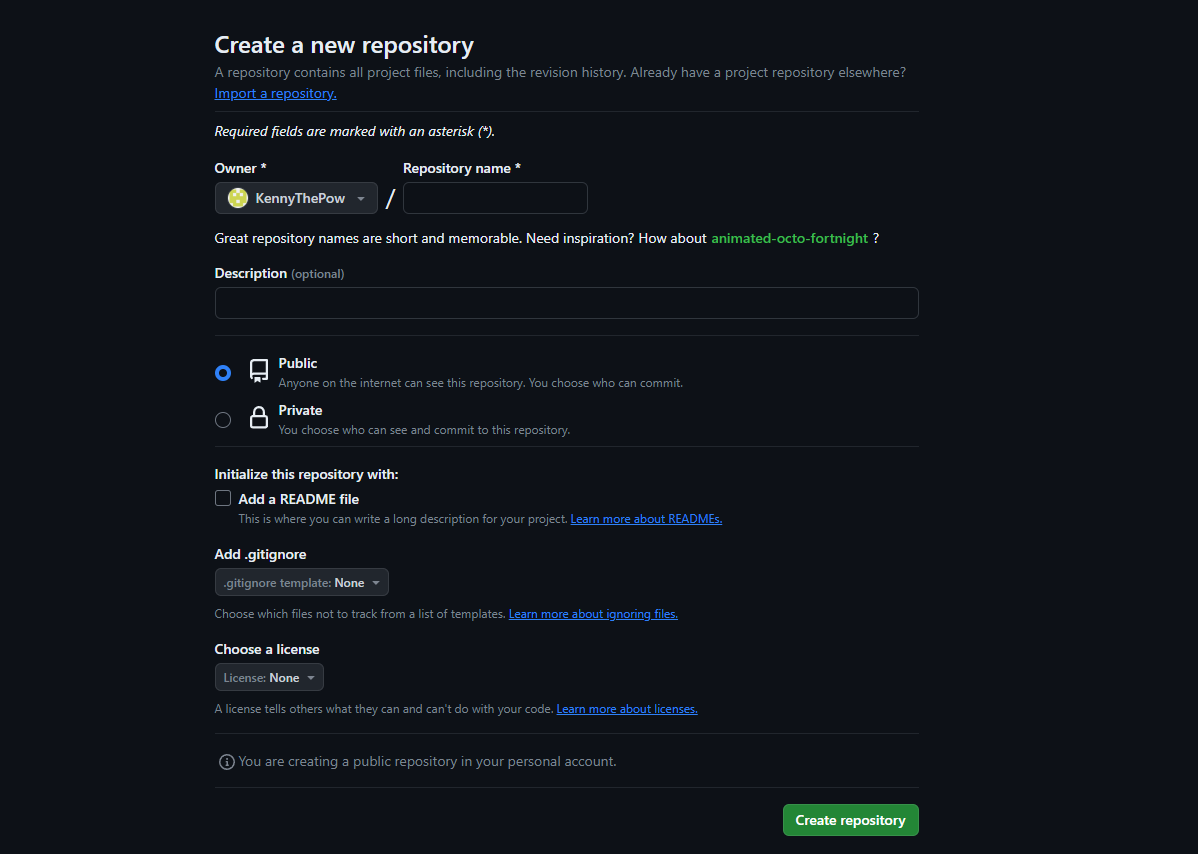
Ukoliko želimo da promenimo neki fajl , možemo to videti i preko Git-a čak i videti razlike između nove verzije fajla i komitovane verzije. Tada dobijamo i opciju da komitujemo novu verziju kao trenutnu ili da ostanemo na staroj.



Slika 4. Command window git diffa i razrešavanja razlike

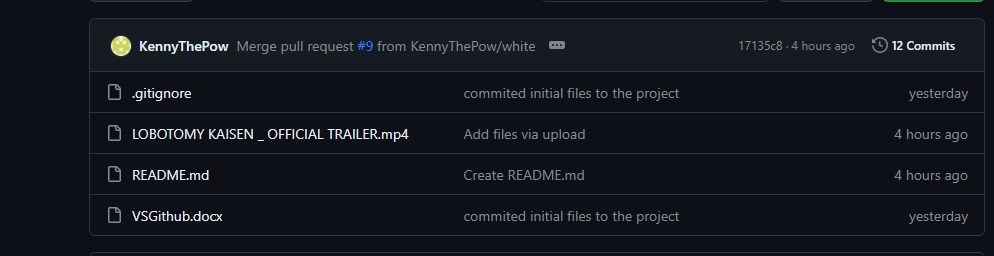
# GitHub deo projekta

Potrebno je prvo napraviti repozitorijum na GitHubu pritiskom dugmeta new, nakon čega je neophodno pratiti uputstva , kao i metod povezivanja prethodno opisanog lokalnog Git repozitorijuma sa GitHub cloud repozitorijumom. Nakon toga, projekat može da počne.



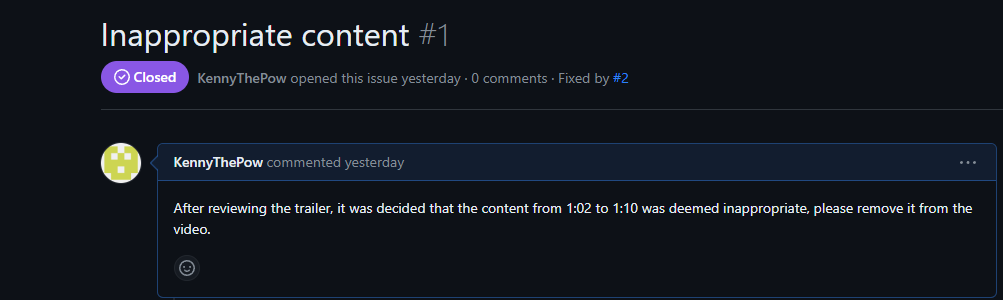
Slika 5. Izgled UI GitHub kreacije projekta

Nakon što je GitHub repozitorijum napravljen, svi prethodno komitovani lokalni Git fajlovi će biti na njemu



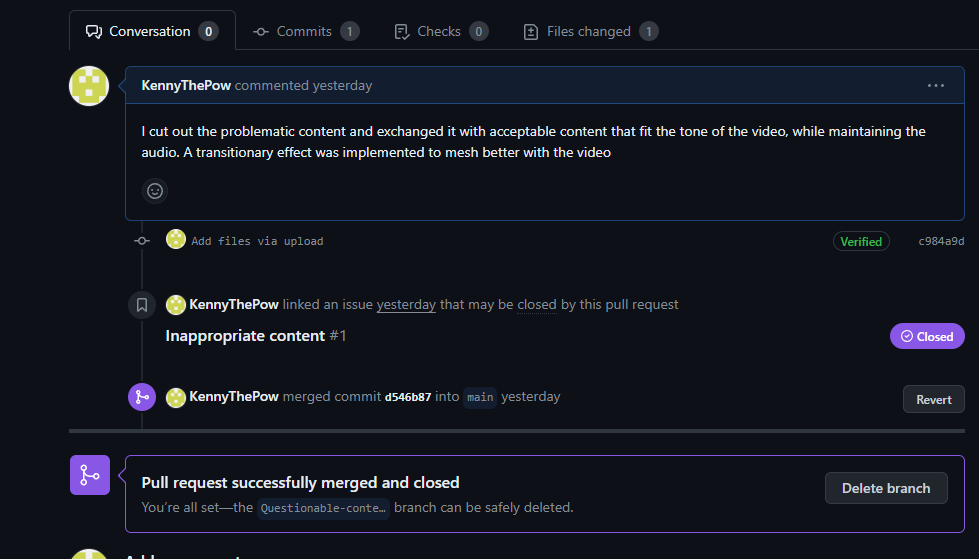
Slika 6. Izled repozitorijuma na GitHub-u

Ovde je zamišljeno da moj projekat zapravo predstavlja grupni rad ekipe koja pravi filmski trejler. Vođa tima obaveštava druge članove o neophodnim izmenama, dodacima i funkcionalnostima koje se trebaju dodati na već postojeći trejler. Prvi takav problem on i navodi korišćenjem prethodno navedene funkcionalnosti praćenja problema. On navodi problem, opisuje traženi ishod i dodeljuje zadatak određenom članu tima.



Slika 7. Izled ISSUE tab-a GitHuba

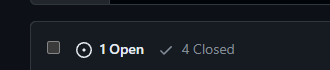
Nakon što član koji je za to zadužen reši problem adekvatnom upotrebom video post produkcijskih alata, on preko Pull zahteva postavlja svoje rešenje uz objašnjenje i time omogućava ostalim članovima da eventualno prodiskutuju pruženu izmeni, ili ovlašćenim članovima da usvoje tu izmenu. On zapravo stvara novu granu projekta, koja sadrži sve što i stari projekat, ali sa datom izmenom. Nakon odobrenja izmene, ta grana spaja se u glavnu granu i može se preko GitHub-a vrlo lako odstraniti, dok je u Git-u neophodno iskucati neophodne komande u kontrolnoj liniji.



Slika 8. Izgled interfejsa pull zahteva

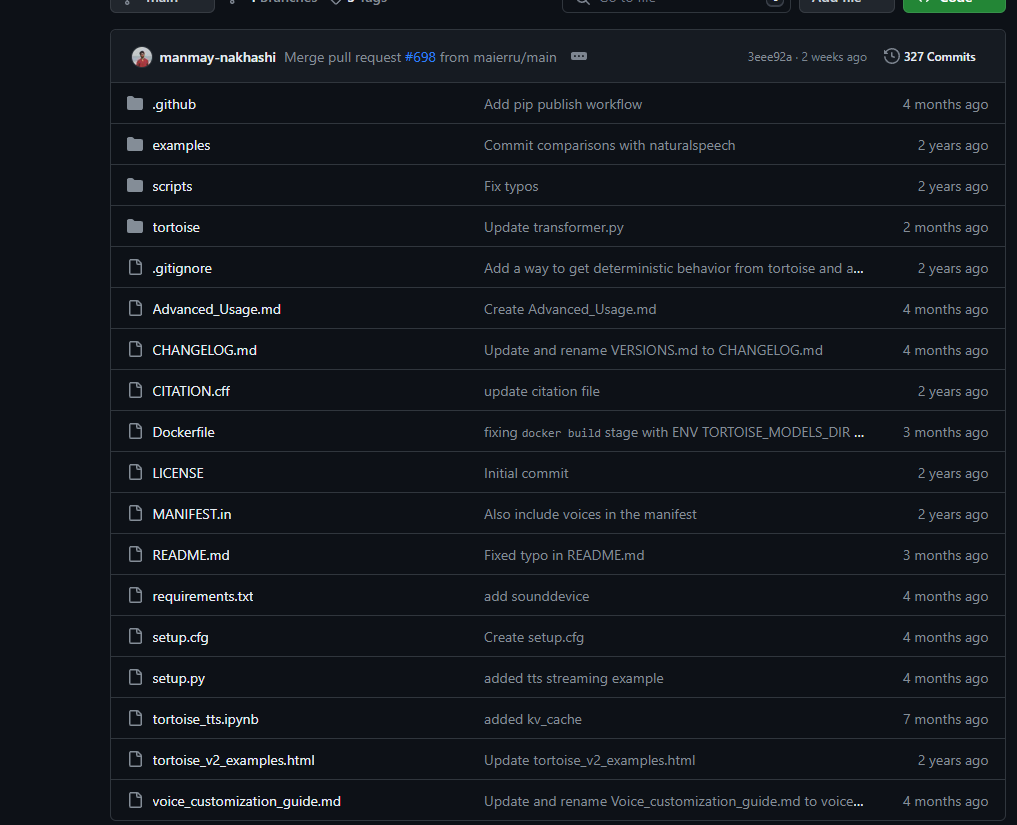
Data grana može se lako vratiti nakon brisanja, kako preko Gita tako i preko Githuba.

Jasno se vidi da opisanim procesom dolazi do vrlo efikasne i lake saradnje u jednom video projektu. U mom, datom primeru ovakav metod rada se nastavlja u još nekih 5 datih zahteva i 5 datih izmena na trejleru.



Slika 9. Istorija issue zahteva

Bitno je osvrnuti se na to da je GitHub zapravo jedna open source zajednica. Primera radi, u mom pokaznom video projektu, potrebno je napraviti AI generisan glasovni sadržaj za potrebe filmskog trejlera. GitHub omogućava lak pristup jednom vrlo jakom open source text to speech programu proizvedenom upravo na GitHubu. Naš video projekat sada zahvaljujući GitHubu ima mogućnost kreiranja vrlo dobrog autoregresivnog modela koji nam pomaže da generišemo dati glasnovni sadržaj. Ovim se ističe jedna vrlo značajna korist GitHuba u video projektima a to je mogućnost korišćenja mnogih koristi koje pružaju već postojeći open source projekti , za koje bi nam trebalo znatno vreme, ali i novac da bi im inače pristupili.

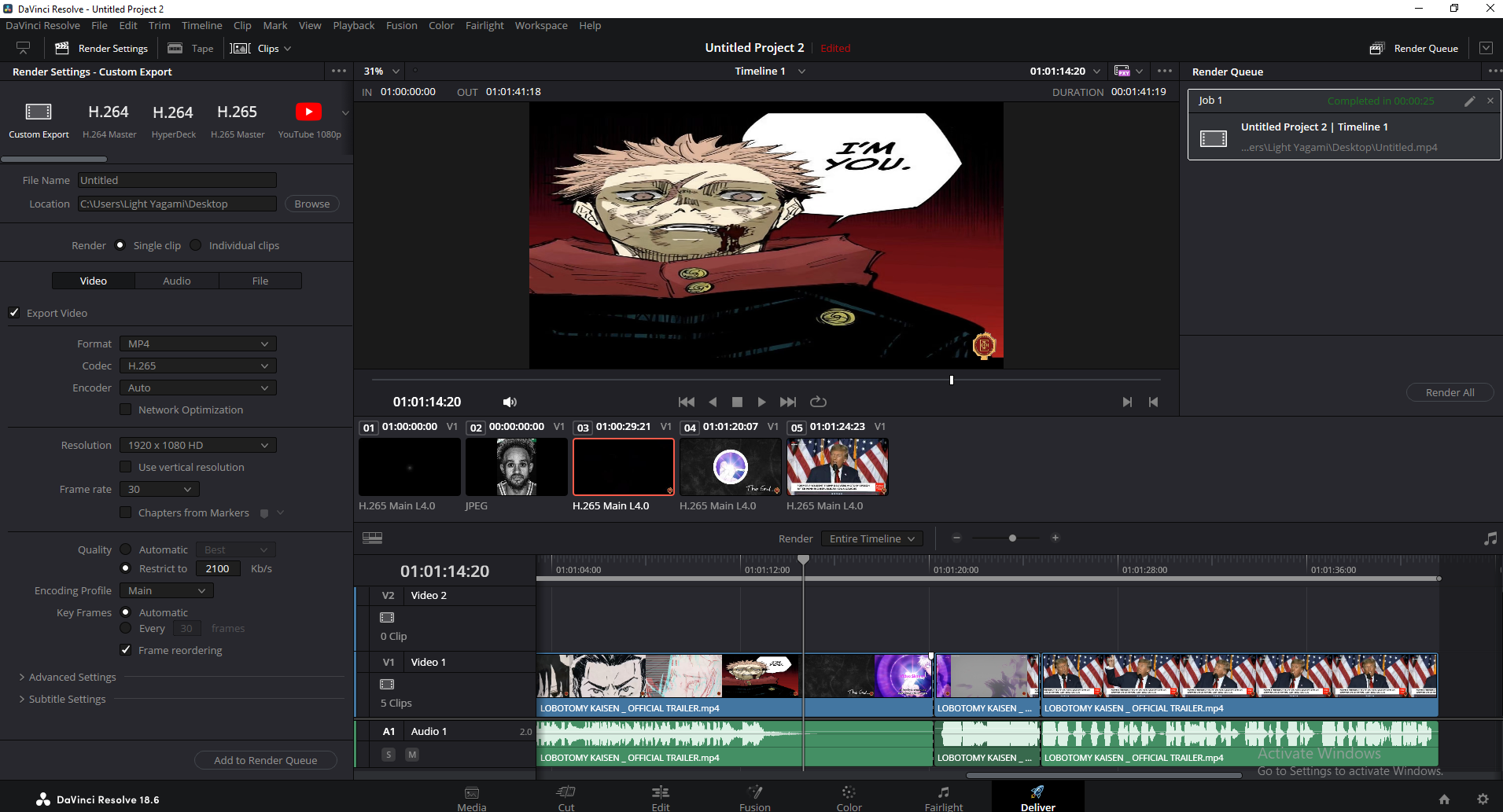


Slika 10. Izgled repozitorijuma tortoise tts aplikacije

Dati program korišćen je uz upotrebu pytorch i anaconda okruženja.

Naravno, GitHub, a samim tim i Git, ne omogućavaju neposrednu video post produkciju. Kao što je opisano, oni omogućavaju kontrolu verzije, organizaciju, pa čak i moćnu biblioteku open source aplikacija kao što smo videli, ali za samu video post produkciju i editing moramo koristiti softver koji je neophodan za to, tu nam Git ne može pomoći u izradi video projekta. Važno je razumeti koristi ali i limitacije svakog alata.

Ovde sam za potrebe pokaznog projekta koristio prethodno napomenuti DaVinci Resolve za potrebe same video montaže. Kako su zadavani zadaci preko GitHub issues , tako sam ja tražene revizije edit-ovau u datom softveru. Izgled ovog okruženja prikazan je na sledećoj slici.



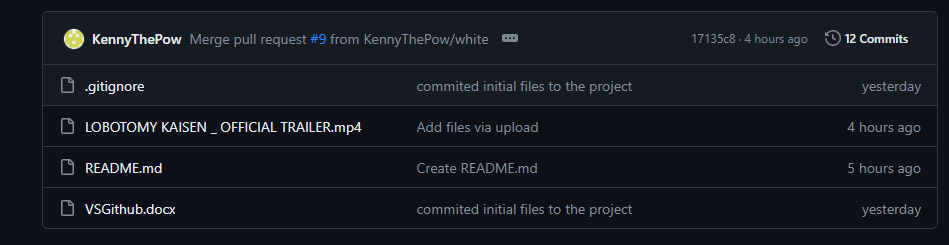
Slika 11. Izgled korisničkog interfejsa DaVinci Resolve-a

Jasno se vide sve odlike jednog video editor okruženja. Vidimo šinu na kojoj su jasno podeljeni audio i video segmenti, vidimo razne vrste efekata kao i alata , standarda snimanja video snimka itd. Mogućnosti ovog okruženja daleko prevazilaze ono što bih ja mogao da navedem, jer DaVinci Resolve praktično postaje standard video editora te ima sve očekivane funkcionalnosti.

Napominjem da je većina editinga za ovaj projekat urađena u skladu sa prezentacijom broj 7 sa predmeta Video Sistemi.

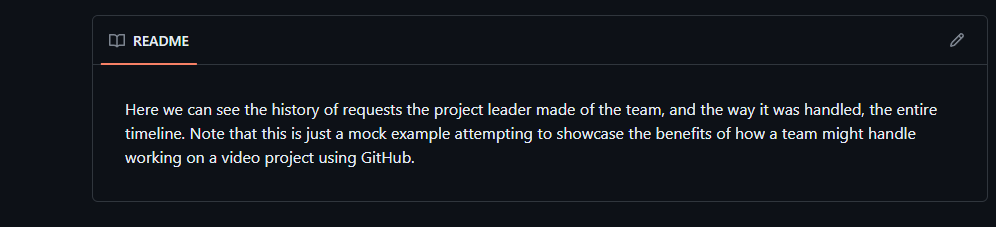
Vrlo važna olakšica koju nam Git pruža u ovakvom jednom video projektu je to što ukoliko novija verzija videa ne valja (a nekako je prošla u main verziju a da nije odbijena), lako možemo vratiti prethodnu, ispravnu verziju. To je još jedan važan doprinos koji Git daje jednom video projektu.

Svaki fajl u datom GitHub repozitorijumu pokazuje kada je poslednji put ažuriran, ko ga je ažurirao kao i istorijat:



Slika 12. Konačan izgled repozitorijuma

Napominje se da ipak nije bilo moguće demonstrirati koliko zapravo Git i Github olakšavaju jedan video projekat, iz razloga što je samo jedna osoba radila na projektu, stoga nije moguće adekvatno demonstirati socijalni aspekt GitHuba, kao ni organizacioni aspekt. Ipak, nadam se da sam uspeo da predstavim mnoge olakšice koje ovaj program obezbeđuje i njegovu korist u polju video projekata.



Slika 13. README sekcija GitHub-a

# Zaključak

Git je vredan alat koji treba uzeti u obzir u video projektima za timove koji rade zajedničku video produkciju. Njegova distribuirana kontrola verzije zajedno sa promišljenim granjanjem i kolaboraciom povećava efikasnost i organizaciju celog workflow-a. Mogućnost eksperimentisanja različitim vrstama editovanja koje posle možemo smestiti u grane, isticanje bitnih dostignuća, ili održavanja čitkog i urednog repozitorijuma znatno doprinose jasnom i transparentnom procesu video editinga. Zajednički rad različitih članova tima postaje trivijalan, posebno kada rade kroz različite grane i onda ih spajaju po potrebi.Povrh toga, GitHub produžuje kolaboracione mogućnosti time što nudi mogućnosti kao praćenje problema, pull zahteve i sve ostale alatke za rukovođenje projektom.Korišćenjem Gita, timovi koji rade na video projektima ne samo da mogu da drže evidenciju promena i verzija na jedan efikasasn način, već i da kreiraju kolaborativnu sredinu koja podstiče komunikaciju, ubrzava razvoj i vodi ka uspešnoj realizaciji kreativne vizije sveta video produkcije, post produkicje i editinga.

Povrh toga, kao što smo videli, GitHub se može efikasno iskoristiti kao jedan resurs gde možemo iskoristiti druge open source projekte kako bi dopunili svoj video projekat. GitHub kao jedna globana platforma privlači velik broj vrlo zanimljivih i korisnih projekata koje možemo iskoristiti da postignemo nešto što teško da bi bilo moguće bez njega. Projekti poput korišćenog tortoise tts mogu i te kako doprineti svakom video projektu i time obezbediti znatno širi opseg mogućnosti.

# Literatura

* *Video sistemi prezentacije- Ana Gavrovska*
* *Pro Git Scott Chacon, Ben Straub*
* *Chacon & Straub 2014*
* [*https://github.com/neonbjb/tortoise-tts*](https://github.com/neonbjb/tortoise-tts)
* *"Git – Distributed Workflows". Git.*