|  |
| --- |
| Topic : Git and Github assignment. |
|  |  |
|  | Use GIT as local VCS. Steps to follow: |
|  | 1. Create a directory ‘project\_dir’ & cd to ‘project\_dir’. |
|  | Ans : creating a directory |
|  | mkdir project\_dir |
|  |  |
|  | moving to the directory |
|  | cd project\_dir |
|  | 2. Initialize git version database. |
|  | Ans : git init |
|  |  |
|  | 3. Create a new file index.html. |
|  | Ans : touch index.html |
|  | 4. Check the git status. You should find index.html as untracked file. |
|  | Ans : git status |
|  | 5. Stage the index.html file. |
|  | Ans : git add index.html |
|  | 6. Commit index.html. |
|  | Ans : git commit -m “html file is created” |
|  | 7. Make few changes in index.html & create a new file info.txt file. |
|  | Ans : touch info.txt |
|  | 8. Check git status. You should find index.html & info.txt as untracked files. |
|  | Ans : git status |
|  | 9. Configure GIT to ignore all txt files. |
|  | Ans : touch.gitignore > .gitignore file- “\*.txt” |
|  | 10. Again check the git status. You should find only index.html as untracked file. |
|  | Ans : git status |
|  | 11. State & commit index.html. |
|  | Ans : git add index.html > git commit -m ”Done ” |
|  | 12. Log all your comments so far. |
|  | Ans : git log |
|  | 13. Make some changes in index.html. |
|  | Ans : A more line added. |
|  | 14. Revert the change made in the previous step using git command. |
|  | Ans: git checkout – |
|  | 15. Again change index.html. |
|  | Ans : Added another line. |
|  | 16. Stage index.html. |
|  | Ans : git add index.html |
|  | 17. Revert back the last stage. |
|  | Ans : git rm --cached index.html |
|  | 18. Rename ‘add’ command to ‘my-add’. |
|  | Ans: alias my-add='git add' |
|  | 19. Using my\_add command Stage index.html again & commit the changes. |
|  | Ans : my-add index.html > git commit -m ”Done”. |
|  | 20. Revert the last commit. |
|  | Ans: git revert HEAD |
|  |  |
|  | GIT BRANCHING |
|  | SECTION-1 (HTML assignments) - Steps to follow: |
|  | 21. First take a backup of your assignments & projects. This is required because due to incorrect GIT operation you may lose your files. |
|  | Ans : Copied the assignment in different folder. |
|  | 22. Create an empty directory ‘Assignments’ & cd to ‘Assignments’. |
|  | Ans : creating a directory |
|  | mkdir Assignments |
|  | moving to the directory |
|  | cd Assignments |
|  | 23. Create a file README.txt inside ‘Assignments’ & write few lines about the contents of ‘Assignments’ folder. |
|  | Ans : touch README.txt |
|  | 24. Commit README.txt file. |
|  | Ans : git add README.txt |
|  | 25. Now create a new branch ‘html-assignments’ |
|  | Ans : git branch html-assignments |
|  | 26. Switch to ‘html-assignments’ branch |
|  | Ans : git checkout html-assignments |
|  | 27. Copy all HTML assignments inside ‘Assignments’ folder. |
|  | Ans : copied |
|  | 28. Commit HTML assignments into ‘html-assignments’ branch. |
|  | Ans : git add index.html > git commit -m "Done” |
|  | 29. Make minor changes into few files belonging to ‘html-assignments’ branch. |
|  | Ans : a line added |
|  | 30. Commit those changed files. |
|  | Ans : git add index.html > git commit -m "Done” |
|  | 31. Switch to master branch. |
|  | Ans : git checkout master |
|  | 32. Make minor changes into README.txt file & commit those changes into master. |
|  | Ans : A line added > git add README.txt > git commit -m “Done” |
|  | 33. Again switch to ‘html-assignments’ branch. |
|  | Ans : git checkout html-assignments. |
|  | 34. Make minor changes into few files belonging to ‘html-assignments’ branch. |
|  | Ans : A new line added. |
|  | 35. Commit those changes. |
|  | Ans : git add index.html > git commit -m "Done" |
|  | 36. Switch to master. |
|  | Ans : git checkout master |
|  | 37. Merge ‘html-assignments’ branch into master. Confirm all html assignments are shown in master. |
|  | Ans : git merge html-assignments > git log --oneline |
|  | 38. Finally delete the ‘html-assignments’ branch. |
|  | Ans : git branch -d html-assignments |
|  |  |
|  | SECTION-2 - (CSS assignments) Steps to follow: |
|  | 1. Create a new branch ‘css-assignments’. |
|  | Ans : git branch css-assignments |
|  | 2. Switch to ‘css-assignments’ branch. |
|  | Ans : git checkout css-assignments |
|  | 3. Copy all CSS assignments inside ‘Assignments’ folder. |
|  | Ans : Copied. |
|  | 4. Commit CSS assignments into ‘css-assignments’ branch. |
|  | Ans : git add style.css > git commit -m "Done” |
|  | 5. Make minor changes into README.txt file on line 1 belonging to ‘css-assignments’ branch. |
|  | Ans: A new line added. |
|  | 6. Commit those changed files. |
|  | Ans : git add README.txt > git commit -m "Comitted" |
|  | 7. Switch to master branch. |
|  | Ans : git checkout master |
|  | 8. Make minor changes into README.txt file on line 3 & commit those changes into master |
|  | Ans : git commit -am "Changes made” |
|  | 9. Again switch to ‘css-assignments’ branch. |
|  | Ans : git checkout css-assignments |
|  | 10. Make minor changes into few files belonging to ‘css-assignments’ branch |
|  | Ans : A newline added |
|  | 11. Commit those changes. |
|  | Ans : git commit -am”Done” |
|  | 12. Switch to master. |
|  | Ans : git checkout master |
|  | 13. Merge ‘css-assignments’ branch into master. Confirm all css assignments are shown in |
|  | master. |
|  | Ans : git merge css-assignments > git commit -am "Done" |
|  | 14. Finally delete the ‘css-assignments’ branch |
|  | Ans : git branch -d css-assignments |
|  |  |
|  |  |
|  | Section 3 |
|  | 1. Create a new branch ‘js-assignments’. |
|  | Ans : git branch js-assignments |
|  | 2. Switch to ‘js-assignments’ branch. |
|  | Ans : git checkout js-assignments |
|  | 3. Copy all JavaScript assignments inside ‘Assignments’ folder. |
|  | Ans : Done |
|  | 4. Commit JavaScript assignments into ‘js-assignments’ branch. |
|  | Ans : git commit -am "A new file is added" |
|  | 5. Make minor changes into README.txt file on line 1 belonging to ‘js-assignments’ branch. |
|  | Ans : Done |
|  | 6. Commit those changed files. |
|  | Ans : git commit -am "Done" |
|  | 7. Switch to master branch. |
|  | Ans : git checkout master |
|  | 8. Make minor changes into README.txt file on line 1 & commit those changes into |
|  | master. |
|  | Ans : DONE > git commit -am "More changes to README.txt from master branch" |
|  | 9. Again switch to ‘js-assignments’ branch. |
|  | Ans : git checkout master |
|  | 10. Make minor changes into few files belonging to ‘js-assignments’ branch. |
|  | Ans : DONE |
|  | 11. Commit those changes. |
|  | Ans : git commit -am "Made more change to the script.js file" |
|  | 12. Switch to master. |
|  | Ans : git checkout master |
|  | 13. Merge ‘js-assignments’ branch into master. Confirm all JavaScript assignments are |
|  | shown in master. |
|  | Ans : git merge js-assignments > git commit -am "Done" |
|  | 14. Finally delete the ‘js-assignments’ branch. |
|  | Ans : git branch -d js-assignments |
|  |  |
|  |  |
|  |  |
|  | Git Remoting |
|  | section 1 |
|  |  |
|  | 39. Create a github account if you do not have already. |
|  | Ans : user name: capgem-training, email: imargs05@gmail.com |
|  | 40. Login on into github account. |
|  | Ans : Done |
|  | 41. Create new public repository ‘freshersbatch-oct16’. |
|  | Ans : Done |
|  | 42. Commit & push any sample file to this repository under ‘Assignments’ directory. |
|  | Ans : git remote add "https://github.com/capgem-training/freshersbatch-oct16.git" > git push origin master |

MOngodb assignment 1

|  |
| --- |
| create database |
|  |  |
|  | Connect to a running mongo instance, use a database named mongo\_practice. use mongo\_practice |
|  |  |
|  | Insert Documents |
|  |  |
|  | Insert the following documents into a movies collection. |
|  |  |
|  | title : Fight Club |
|  | writer : Chuck Palahniuk |
|  | year : 1999 |
|  | actors : [ |
|  | Brad Pitt |
|  | Edward Norton |
|  | ] |
|  | Ans : |
|  |  |
|  | db.movies.insert({title:"Fight Club", writer: "Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]}) |
|  | title : Pulp Fiction |
|  | writer : Quentin Tarantino |
|  | year : 1994 |
|  | actors : [ |
|  | John Travolta |
|  | Uma Thurman |
|  | ] |
|  |  |
|  | db.movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"2009", actors:["John Travolta", "Uma Thurman"]}) |
|  | title : Inglorious Basterds |
|  | writer : Quentin Tarantino |
|  | year : 2009 |
|  | actors : [ |
|  | Brad Pitt |
|  | Diane Kruger |
|  | Eli Roth |
|  | ] |
|  |  |
|  | db.movies.insert({title:"Inglorious Basterds", writer:"Quentin Tarantino", year:"2009", actors:["Brad Pitt", "Diane Kruger", "Eli Roth"]}) |
|  | title : The Hobbit: An Unexpected Journey |
|  | writer : J.R.R. Tolkein |
|  | year : 2012 |
|  | franchise : The Hobbit |
|  |  |
|  | db.movies.insert({title:"The Hobbit: An unexpected Journey", writer:"J.R.R. Tolkein", year:"2012",franchise:"The Hobbit"}) |
|  | title : The Hobbit: The Desolation of Smaug |
|  | writer : J.R.R. Tolkein |
|  | year : 2013 |
|  | franchise : The Hobbit |
|  |  |
|  | db.movies.insert({title:"The Hobbit: The Desolation of Smaug", writer:"J.R.R Tolkien", year:"2013", franchise:"The Hobbit"}) |
|  | title : The Hobbit: The Battle of the Five Armies |
|  | writer : J.R.R. Tolkein |
|  | year : 2012 |
|  | franchise : The Hobbit |
|  | synopsis : Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness. |
|  |  |
|  | db.movies.insert({title:"The Hobbit: The Battle of the Five Armies", writer:"J.R.R Tolkien", year:"2002", franchise:"The Hobbit", synopsis:"Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness."}) |
|  | title : Pee Wee Herman's Big Adventure |
|  |  |
|  | db.movies.insert({title:"Pee Wee Herman's Big Adventures"}) |
|  | title : Avatar |
|  |  |
|  | db.movies.insert({title:"Avatar"}) |
|  | ------------------------------------------------------------------------------------------- |
|  | Query / Find Documents |
|  | query the movies collection to |
|  |  |
|  | 1.get all documents |
|  | db.movies.find() |
|  |  |
|  | 2.get all documents with writer set to "Quentin Tarantino" |
|  | db.movies.find({writer:"Quentin Tarantino"}) |
|  |  |
|  | 3.get all documents where actors include "Brad Pitt" |
|  | db.movies.find({actors:"Brad Pitt"}) |
|  |  |
|  | 4.get all documents with franchise set to "The Hobbit" |
|  | db.movies.find({franchise:"The Hobbit"}) |
|  |  |
|  | 5.get all movies released in the 90s |
|  | db.movies.find({year:{$gt:"1990", $lt:"2000"}}) |
|  |  |
|  | 6.get all movies released before the year 2000 or after 2010 |
|  | db.movies.find({$or:[{year:{$gt:"2010"}},{year: {$lt:"2000"}}]}) |
|  |  |
|  |  |
|  | Update Documents |
|  | 1.add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." |
|  | Ans: |
|  | db.movies.update({\_id:ObjectId("5c9f98e5e5c2dfe9b3729bfe")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}}) |
|  |  |
|  | 2.add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." |
|  | Ans: |
|  | db.movies.update({\_id:ObjectId("5c9fa42ae5c2dfe9b3729c03")}, {$set:{synopsis:"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}}) |
|  | 3.add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction" |
|  | Ans : |
|  | db.movies.update({\_id:ObjectId("5c9f983ce5c2dfe9b3729bfc")}, {$push:{actors:"Samuel L. Jackson"}}) |
|  |  |
|  | Text Search |
|  |  |
|  | 1.find all movies that have a synopsis that contains the word "Bilbo" |
|  |  |
|  | db.movies.find({synopsis:{$regex:"Bilbo"}}) |
|  |  |
|  | 2.find all movies that have a synopsis that contains the word "Gandalf" |
|  |  |
|  | db.movies.find({synopsis:{$regex:"Gandalf"}}) |
|  |  |
|  | 3.find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf" |
|  |  |
|  | db.movies.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]}) |
|  |  |
|  | 4.find all movies that have a synopsis that contains the word "dwarves" or "hobbit" |
|  |  |
|  | db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]}) |
|  |  |
|  | 5.find all movies that have a synopsis that contains the word "gold" and "dragon" |
|  |  |
|  | db.movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]}) |
|  |  |
|  | Delete Documents |
|  |  |
|  | 1.delete the movie "Pee Wee Herman's Big Adventure" |
|  |  |
|  | db.movies.remove({\_id:ObjectId("5c9f992ae5c2dfe9b3729c00")}) |
|  |  |
|  | 2.delete the movie "Avatar" |
|  |  |
|  | db.movies.remove({\_id:ObjectId("5c9f9936e5c2dfe9b3729c01")}) |
|  |  |
|  | Relationships |
|  | Insert the following documents into a users collection |
|  | username : GoodGuyGreg |
|  | first\_name : "Good Guy" |
|  | last\_name : "Greg" |
|  | Ans : |
|  | db.users.insert({\_id:1,username:"GoodGuyGreg", first\_name:"Good Guy", last\_name:"Greg"}) |
|  | username : ScumbagSteve |
|  | full\_name : |
|  | first : "Scumbag" |
|  | last : "Steve" |
|  |  |
|  | db.users.insert({\_id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}}) |
|  |  |
|  | Insert the following documents into a posts collection |
|  | username : GoodGuyGreg |
|  | title : Passes out at party |
|  | body : Wakes up early and cleans house |
|  | db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"}) |
|  | username : GoodGuyGreg |
|  | title : Steals your identity |
|  | body : Raises your credit score |
|  | Ans : |
|  |  |
|  | db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"}) |
|  | username : GoodGuyGreg |
|  | title : Reports a bug in your code |
|  | body : Sends you a Pull Request |
|  |  |
|  | db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"}) |
|  | username : ScumbagSteve |
|  | title : Borrows something |
|  | body : Sells it |
|  |  |
|  | db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"}) |
|  | username : ScumbagSteve |
|  | title : Borrows everything |
|  | body : The end |
|  |  |
|  | db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"}) |
|  | username : ScumbagSteve |
|  | title : Forks your repo on github |
|  | body : Sets to private |
|  |  |
|  | db.posts.insert({username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"}) |
|  |  |
|  | Insert the following documents into a comments collection |
|  |  |
|  | username : GoodGuyGreg |
|  | comment : Hope you got a good deal! |
|  | post : [post\_obj\_id] |
|  | where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something" |
|  |  |
|  | db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("5ca0b7e96435f98b5901f463")}) |
|  |  |
|  | username : GoodGuyGreg |
|  | comment : What's mine is yours! |
|  | post : [post\_obj\_id] |
|  | where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything" |
|  |  |
|  | db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("5ca0b9706435f98b5901f46a")}) |
|  |  |
|  | username : GoodGuyGreg |
|  | comment : Don't violate the licensing agreement! |
|  | post : [post\_obj\_id] |
|  | where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github |
|  |  |
|  | db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("5ca0b8766435f98b5901f467")}) |
|  |  |
|  | username : ScumbagSteve |
|  | comment : It still isn't clean |
|  | post : [post\_obj\_id] |
|  | where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party" |
|  |  |
|  | db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("5ca0b8546435f98b5901f466")}) |
|  |  |
|  | username : ScumbagSteve |
|  | comment : Denied your PR cause I found a hack |
|  | post : [post\_obj\_id] |
|  | where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code" |
|  |  |
|  | db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("5ca0b9256435f98b5901f469")}) |
|  |  |
|  |  |
|  | Querying related collections |
|  | 1.find all users |
|  |  |
|  | db.users.find().pretty() |
|  |  |
|  | 2.find all posts |
|  |  |
|  | db.posts.find().pretty() |
|  | 3.find all posts that was authored by "GoodGuyGreg" |
|  |  |
|  | db.posts.find({username:"GoodGuyGreg"}) |
|  |  |
|  | 4.find all posts that was authored by "ScumbagSteve" |
|  |  |
|  | db.posts.find({username:"ScumbagSteve"}) |
|  |  |
|  | 5.find all comments |
|  |  |
|  | db.comments.find().pretty() |
|  |  |
|  | 6.find all comments that was authored by "GoodGuyGreg" |
|  |  |
|  | db.comments.find({username:"GoodGuyGreg"}) |
|  |  |
|  | 7.find all comments that was authored by "ScumbagSteve" |
|  |  |
|  | db.comments.find({username:"ScumbagSteve"}) |
|  |  |
|  | 8.find all comments belonging to the post "Reports a bug in your code" |
|  |  |
|  | db.posts.aggregate([ |
|  | { |
|  | $match: { title: 'Reports a bug in your code' } |
|  | }, |
|  | { |
|  | $lookup: { |
|  | from: 'comments', |
|  | localField: '\_id', |
|  | foreignField: 'post', |
|  | as: 'comments' |
|  | } |
|  | } |
|  | ]).pretty(); |

Satyam dev github assignment

|  |
| --- |
| --------------------------------------Assignment 1 -------------------------------------- |
|  |  |
|  | 1. Create a directory ‘project\_dir’ & cd to ‘project\_dir’. |
|  | => mkdir project\_dir |
|  | cd project\_dir/ |
|  | 2. Initialize git version database. (git init) |
|  | => git init |
|  | 3. Create a new file index.html. |
|  | => touch index.html |
|  | 4. Check the git status. You should find index.html as untracked file. |
|  | => git status |
|  | 5. Stage the index.html file. |
|  | => git add index.html |
|  | 6. Commit index.html |
|  | => git commit -m "Created index.html" |
|  | 7. Make few changes in index.html & create a new file info.txt file. |
|  | => touch info.txt |
|  | 8. Check git status. You should find index.html & info.txt as untracked files. |
|  | => git status |
|  | 9. Configure GIT to ignore all txt files. |
|  | => touch .gitignore |
|  | => add the following line to the .gitnore file- "\*.txt" |
|  | 10. Again check the git status. You should find only index.html as untracked file. |
|  | => git status |
|  | 11. State & commit index.html |
|  | => git add index.html |
|  | => git commit -m "Added emmet abbreviation to index.html and added all .txt files to gitignore" |
|  | 12. Log all your comments so far. |
|  | => git log |
|  | 13. Make some changes in index.html. |
|  | => Added Header H1 |
|  | 14. Revert the change made in the previous step using git command. |
|  | => git checkout -- . |
|  | => since the changes were not commited the above command reverted the changes back to the previous commit |
|  | 15. Again change index.html. |
|  | => Added a paragraph |
|  | 16. Stage index.html |
|  | => git add index.html |
|  | 17. Revert back the last stage. |
|  | => git rm --cached index.html |
|  | 18. Rename ‘add’ command to ‘my-add’. |
|  | => alias my-add='git add' |
|  | 19. Using my\_add command Stage index.html again & commit the changes. |
|  | => my-add index.html |
|  | => git commit -m "New change with error that has to be reverted" |
|  | 20. Revert the last commit. |
|  | => git revert HEAD |
|  |  |
|  |  |
|  | --------------------------------------Assignment 2: Section 1 -------------------------------------- |
|  | 21. First take a backup of your assignments & projects. This is required because due to incorrect GIT operation you may lose your files. |
|  | => copy "WebApp" D:\ADAPT\GIT |
|  | 22. Create an empty directory ‘Assignments’ & cd to ‘Assignments’. |
|  | => mkdir Assignments |
|  | => cd Assignments |
|  | => git init |
|  | 23. Create a file README.txt inside ‘Assignments’ & write few lines about the contents of ‘Assignments’ folder. |
|  | => touch README.txt |
|  | 24. Commit README.txt file. |
|  | => git add README.txt |
|  | => git commit -m "Added README file" |
|  | 25. Now create a new branch ‘html-assignments’. |
|  | => git branch html-assignments |
|  | 26. Switch to ‘html-assignments’ branch. |
|  | => git checkout html-assignments |
|  | 27. Copy all HTML assignments inside ‘Assignments’ folder. |
|  | => Done |
|  | 28. Commit HTML assignments into ‘html-assignments’ branch. |
|  | => git add index.html |
|  | => git commit -m "Commiting index.html to the html-assignments branch" |
|  | 29. Make minor changes into few files belonging to ‘html-assignments’ branch. |
|  | => Made changes |
|  | 30. Commit those changed files. |
|  | => git add index.html |
|  | => git commit -m "Made minor changed to index.html" |
|  | 31. Switch to master branch. |
|  | => git checkout master |
|  | 32. Make minor changes into README.txt file & commit those changes into master. |
|  | => Made changes |
|  | => git add README.txt |
|  | => git commit -m "Made changes in the readme file" |
|  | 33. Again switch to ‘html-assignments’ branch. |
|  | => git checkout html-assignments |
|  | 34. Make minor changes into few files belonging to ‘html-assignments’ branch. |
|  | => Made more changes |
|  | 35. Commit those changes. |
|  | => git add index.html |
|  | => git commit -m "Made more changes" |
|  | 36. Switch to master. |
|  | => git checkout master |
|  | 37. Merge ‘html-assignments’ branch into master. Confirm all html assignments are shown in master. |
|  | => git merge html-assignments |
|  | => git log --oneline |
|  | 38. Finally delete the ‘html-assignments’ branch. |
|  | => git branch -d html-assignments |
|  |  |
|  |  |
|  |  |
|  | --------------------------------------Assignment 2: Section 2 -------------------------------------- |
|  | 1. Create a new branch ‘css-assignments’. |
|  | => git branch css-assignments |
|  | 2. Switch to ‘css-assignments’ branch. |
|  | =>git checkout css-assignments |
|  | 3. Copy all CSS assignments inside ‘Assignments’ folder. |
|  | => Done |
|  | 4. Commit CSS assignments into ‘css-assignments’ branch. |
|  | => git add style.css |
|  | => git commit -m "Commiting style.css to the css-assignments branch" |
|  | 5. Make minor changes into README.txt file on line 1 belonging to ‘css-assignments’ branch. |
|  | => made changes |
|  | 6. Commit those changed files. |
|  | => git add README.txt |
|  | => git commit -m "Made changes to README.txt from css-assignments branch" |
|  | 7. Switch to master branch. |
|  | => git checkout master |
|  | 8. Make minor changes into README.txt file on line 3 & commit those changes into |
|  | master. |
|  | => made changes |
|  | => git commit -am "Changes made to README" |
|  | 9. Again switch to ‘css-assignments’ branch. |
|  | =>git checkout css-assignments |
|  | 10. Make minor changes into few files belonging to ‘css-assignments’ branch. |
|  | => Done |
|  | 11. Commit those changes. |
|  | => git commit -am "Changes made to style.css" |
|  | 12. Switch to master. |
|  | => git checkout master |
|  | 13. Merge ‘css-assignments’ branch into master. Confirm all css assignments are shown in |
|  | master. |
|  | => git merge css-assignments |
|  | =>This will give error, then open the file that creates conflict and then make changes in the file. |
|  | => git commit -am "Conflicts resolved" |
|  | 14. Finally delete the ‘css-assignments’ branch |
|  | => git branch -d css-assignments |
|  |  |
|  |  |
|  | --------------------------------------Assignment 2: Section 3 -------------------------------------- |
|  | 1. Create a new branch ‘js-assignments’. |
|  | => git branch js-assignments |
|  | 2. Switch to ‘js-assignments’ branch. |
|  | => git checkout js-assignments |
|  | 3. Copy all JavaScript assignments inside ‘Assignments’ folder. |
|  | => Done |
|  | 4. Commit JavaScript assignments into ‘js-assignments’ branch. |
|  | => git commit -am "New file script.js is added" |
|  | 5. Make minor changes into README.txt file on line 1 belonging to ‘js-assignments’ branch. |
|  | => Done |
|  | 6. Commit those changed files. |
|  | => git commit -am "made changes to the README file from js-assignments branch" |
|  | 7. Switch to master branch. |
|  | => git checkout master |
|  | 8. Make minor changes into README.txt file on line 1 & commit those changes into |
|  | master. |
|  | => DONE |
|  | => git commit -am "More changes to README.txt from master branch" |
|  | 9. Again switch to ‘js-assignments’ branch. |
|  | => git checkout master |
|  | 10. Make minor changes into few files belonging to ‘js-assignments’ branch. |
|  | => DONE |
|  | 11. Commit those changes. |
|  | => git commit -am "Made more change to the script.js file" |
|  | 12. Switch to master. |
|  | =>git checkout master |
|  | 13. Merge ‘js-assignments’ branch into master. Confirm all JavaScript assignments are |
|  | shown in master. |
|  | => git merge js-assignments |
|  | =>This will give error, then open the file that creates conflict and then make changes in the file. |
|  | => git commit -am "Conflicts resolved" |
|  | 14. Finally delete the ‘js-assignments’ branch. |
|  | => git branch -d js-assignments |
|  |  |
|  |  |
|  |  |
|  | --------------------------------------Assignment 3: Section 1 -------------------------------------- |
|  | 39. Create a github account if you do not have already. |
|  | => Done |
|  | 40. Login on into github account. |
|  | => Done |
|  | 41. Create new public repository ‘freshersbatch-oct16’. |
|  | => Done |
|  | 42. Commit & push any sample file to this repository under ‘Assignments’ directory. |
|  | => git remote add origin https://github.com/satyamdeb12/freshersbatch-oct16 |
|  | => git push origin master |
|  |  |
|  | --------------------------------------Assignment 3: Section 2 -------------------------------------- |
|  | 1. One developer from project team will create eclipse projects ‘SampleProj’ & add sample |
|  | source code files. Then commit all files through eclipse GIT plugin. |
|  | 2. Collaborate other team members with your github account so that they can also modify |
|  | the committed files. |
|  | 3. Other developers from same team will checkout all files from remote repository. This |
|  | might get conflicts since certain files fail to merge. In such case, merge it manually. |
|  | 4. Commit & push the ‘SampleProj’ project. |