

# **Spring2019 CS5551: Advanced Software Engineering**

Department of Computer Science Electrical Engineering  
University of Missouri - Kansas City

## **Version Control system for Deep Learning (DLV)**

### **Team 6:**

Sindhusha Tiyyagura (24)  
Pradeepika Kolluru (12)  
Sravan kumar Pagadala (21)  
Dinesh kumar Reddy Kusam (14)

#### **GITHUB URL:**

<https://github.com/sindhusha-t/VCS-Git-for-deep-learning>

#### **VIDEO URL:**

<https://www.youtube.com/watch?v=p0Z6mslp40o>

#### **PPT URL:**

[https://drive.google.com/open?id=15k\\_qQV8vwhH7z8nJcK1RgbNCV03Ee\\_KC](https://drive.google.com/open?id=15k_qQV8vwhH7z8nJcK1RgbNCV03Ee_KC)

#### **ZENHUB URL:**

<https://github.com/sindhusha-t/VCS-Git-for-deep-learning#workspaces/vcs-for-deep-learning-5c60d48784111d4401fa452b/board?repos=169820599>

# CONTENTS

- I. [Introduction](#)
- II. [Project Goal](#)
  - a. [Motivation](#)
  - b. [Technologies Used](#)
  - c. [System Architecture](#)
  - d. [System Features](#)
  - e. [Sequence Diagram](#)
- III. [Design](#)
- IV. [Project Plan](#)
- V. [Project Increment 1](#)
  - a. [Tasks Done](#)
  - b. [Design and Implementation](#)
- VI. [Project Increment 2](#)
  - a. [Tasks Done](#)
  - b. [Design and Implementation](#)
- VII. [Project Increment 3](#)
  - a. [Tasks Done](#)
  - b. [Design and Implementation](#)
- VIII. [Project Increment 4](#)
  - a. [Tasks Done](#)
  - b. [Design and Implementation](#)
- IX. [Working of Project](#)
- X. [Future Work](#)
- XI. [References](#)
- XII. [Project Deployment](#)
- XIII. [Project Management](#)

## I. INTRODUCTION:

DLV – Deep learning version control system – Web Application

Where users can register and login to their accounts

To view list of their projects,

To view different versions of experiments done in the model,

To easily view the changes done locally,

To easily commit particular experiment.

## II. PROJECT GOAL:

### MOTIVATION:

The main aim of the project is to develop a version control system for deep learning models (source code, input and output data files, metrics about the experiment) similar to GIT.

- There is no functionality in GIT to track group of files as a version (Deep learning is more dependent on the tracking of each experiment as a version)
- The following tasks can be done with the existing GIT functionalities like **tags and branches**. But this is a very painful task where user has to remember that each experiment need to be tagged / given a branch.
- There is no functionality in GIT to compare the results of different experiments (this means the comparisons for different tags).

### TECHNOLOGIES USED:

Angular

ExpressJS

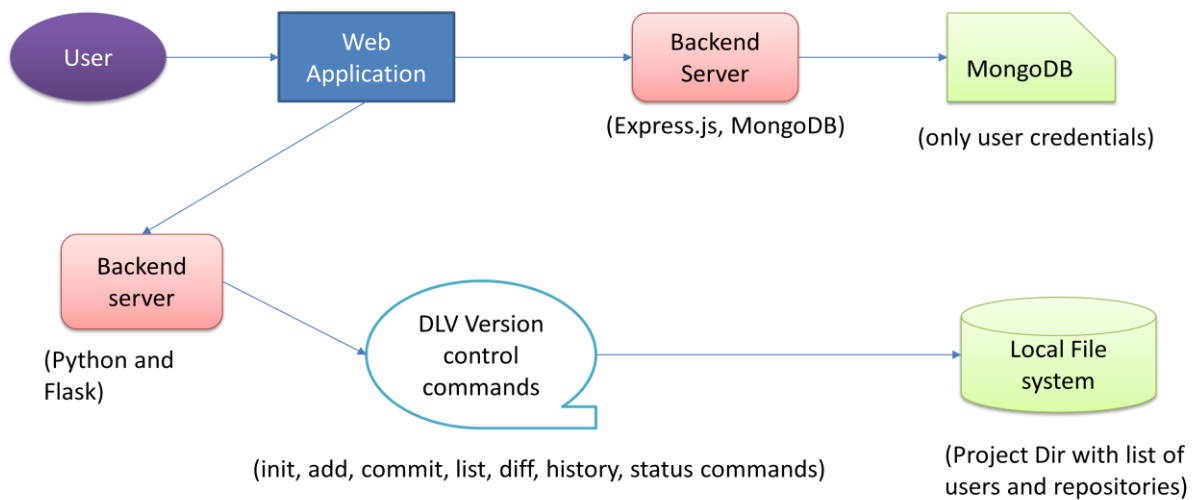
MongoDB

NodeJS

Python and Flask

Heroku

## SYSTEM ARCHITECTURE:



## SYSTEM FEATURES:

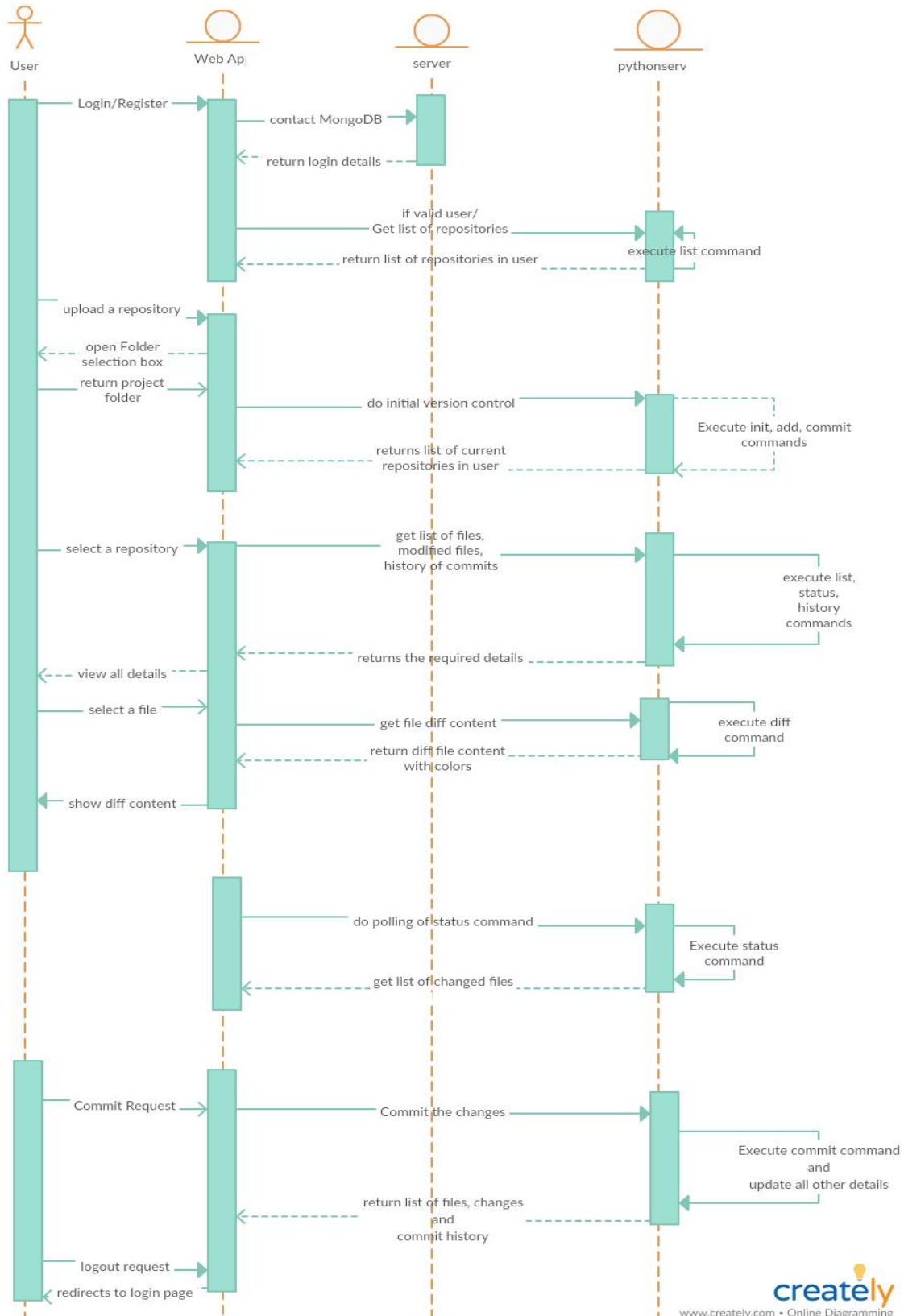
### Command Line Interface for dlv commands:

- dlv init
- dlv add
- dlv commit
- dlv history
- dlv status
- dlv list
- dlv desc
- dlv diff
- dlv push

### User Interface (Integrating the commands with UI):

- User Registration and Login
- User can view list of projects/deep learning models
- User can upload and start versioning the model.
- User can view the changes done locally in the system and verify the changes done before committing the experiment.
- User can view history of all experiments and the changes in it.
- User can view the current version files/ last experiment that is saved.

## SEQUENCE DIAGRAM:



### III. DESIGN:

#### Existing GIT functionality

1. Git tracks each file history independently.
  - Most of the git commands are used to change the git meta data files on the local machine.
  - Only 2 commands( git pull and push commands) are connected to the GIT server for downloading and uploading files respectively using File transfer protocol.
2. GIT init command: Creates .git directory (contains meta data about the files to be tracked) on the project directory.

Example: .git/

- HEAD
- Objects/
  1. refs/
  2. heads/
  3. tags/
- Config
- .gitignore

HEAD → specifies to which branch GIT to commit

config → contains the settings like repository name, user profiles.

objects → directory contains 3 different objects like commit, tree and blob objects.

All these files will be stored in the form of hash file\_name

1. **GIT add command:** It creates cache of all the files specified and creates BLOB objects for all the files.
  - BLOB objects are nothing but the physical files which contains the file content.
  - The BLOB object is not stored with the file name. It will be stored with the md5\_hash value as the file name and with the first two chars as the directory name.
  - MD5 hash value will be same for same file content files. ( different if there is any change in the file contents ).

- The functionality of MD5 checksum filename is to track different versions of the same file. (in which same file\_name cannot be used for tracking different versions ).

Example:

git add file1.py

- .git/
  - objects/
    - 1. info/
      - 1. 92/
        - 1. f65tbjh784hj345j34hmw4

--> BLOB object contains file1.py contents

- It also adds the files to the staging directory which will be later used by commit to commit the files

2) **GIT commit command:** It creates DAG (Directed acyclic graph ) for the list of files committed in the project repository.

- In commit command, it starts tracking the files(BLOB objects) stored in the cache.
- It creates commit and tree objects.  
TREE object basically represents a directory. It references to TREE and BLOB objects.( basically sub directories and files inside the directory ).
- COMMIT object points out to the TREE object(latest committed version) and the parent COMMIT object( previous COMMIT object).

-----> TREE object <-----

Author: Name

commit time: time

TREE 5437hg4t7854th45y45y9y568

BLOB dg67w45g4ry754hg587th458t

BLOB fg478yt57th58th58t5t8945fb

-----> COMMIT Object <-----

Author: Name

commit Time: time

TREE fgdsyu74htfw38eduhw8e3wd8  
PARENT g47uygr4ewiuh4yturhut74e

## Initial Proposed functionality for Deep learning experiments

Will go over the internal working of the 3 commands (dlv init, add and commit ).

### 1. **DLV init command:**

This command is similar to the git init command.

It creates .dlv directory which specifies the metadata for the project repository.

- .dlv/
  - Config
  - cache/
  - HEAD
  - Stage

### 2. **DLV add command:**

This command is also similar to the git add command which creates BLOB objects for the files to be tracked using DLV.

Example:

dlv add file1.py

- .git/
  - cache/
    - 1. 92/
      - 1. f65tbjh784hj345j34hmw4
        - 1. --> BLOB object contains file1.py contents

### 3. **DLV commit command:**

This command is little different compared to the git commit command.

In this it creates TREE and COMMIT objects similar to the git functionality but the structure of the COMMIT object is different compared to the git functionality.

-----> TREE object <-----

Author: Name

commit time: time



TREE 5437hg4t7854th45y45y9y568 # represents SUB\_DIRECTORY  
BLOB dg67w45g4ry754hg587th458t # represents a file  
BLOB fg478yt57th58th58t5t8945fb

-----> COMMIT Object <-----

Author: Name

commit Time: time

code: file1.py

md5: fgf785tg578hg54t8h43wefre34

input: data1.json

md5: df46rgf64h4e8ty58t43efedw43t

output: output\_data.json

md5: gf54gt754ht7854h58t54tr5ty5r

metrics: metrics.json

md5: gf487tg78e5tgf74yty85t5t8y54

commit object contains full details of the experiment run and its versions used in it.

## Final Proposed Design for Deep Learning Version Control System:

- For the previous design – we were unable to test whether the commands are working correctly or not because of the hashing values.
- So during this period we have slightly modified the design.
- The folder structure of .dlv is almost same.
- Below is the final implemented functionality/design.

Example:

- .dlv/
  - Config.txt
  - HEAD.txt
  - Master/
    - Status.txt
    - Commit\_log.txt
    - Commits/
      - Commit.1
      - Commit.2
      - Commit.3
    - Stage/
      - File1
      - File 2

- Cache/
  - File 1.1
  - File 1.2
  - File 2.1
- Branch1/

For each branch that user creates – a new folder is created in the .dlv directory.

## IV. PROJECT PLAN:

### Increment-1:

- Getting hands on with GIT source code.
  - Download git source code
  - Compile and run the code
  - Change one of the module to get the feel of working with GIT
- Understanding the design and coming with the rough design for the project
  - The main difference between GIT version control system and the proposed system is in the functionality of tracking the files.
  - In proposed system, modeling artifacts(models, data sets, source code) need to tracked as a single version, where as in GIT each files is tracked separately.
  - Coming up with a rough design for the new proposed project.

### Increment-2:

- 1) Database to store credentials and large data files.
  - We will be using MongoDB to store user credential details.
  - We will be using local file system to store the large data files and their lineages.
- 2) User registration and authentication/login requests to be handled.
  - Developing scripts to register a user and to authenticate the user.

- Should first connect to DB cache details, if there is a miss response then should connect to the proxy DB for user credentials.
- Enabling the scripts to work in the command line interface.

### **Design Flow:**

1. Client either requests for authentication or login to the server.
2. Server connects to back-end database either to store or retrieve user details.
3. Checks validation and sends response to the client.

### **Increment-3:**

- 1) Developing commands that reflect changes in the local repository (client) without contacting server.
  - Commands that reflect changes only at the client side ( AKA client side scripts/commands ) gitdl init gitdl add gitdl remove gitdl commit gitdl tag gitdl branch gitdl checkout
  - All these commands need to be changed to track artifacts as a single version. The functionality to be followed is similar to the DAG(Directed Acyclic Graph) as of now. ( which includes objects and references to the blob files)
- 2) Developing commands that client connects to the server for the response.
  - Commands that are made changes in the local repository are reflected at the server side. gitdl clone gitdl push gitdl pop
  - Need to think of where to store the large files at the server side.
- 3) Giving the list of repositories in the user workspace.

### **Increment-4:**

- 1) Handling merging, rebasing, and diffing of set of files/ artifacts.
  1. Adding the flexibility of adding and managing branches for the client.
  2. Using tools for merging and diffing of file contents.
- 2) Testing different functionalities in the project

## V. PROJECT INCREMENT 1:

### TASKS DONE:

1. Getting familiar with GIT open source code.
2. Understanding the requirements and coming up with the design for the functionality of dlv commands.
3. Implementing dlv init, add and commit commands.

### DESIGN and IMPLEMENTATION:

Attaching the Screenshots of each command execution

#### 1) DLV INIT:

Just like GIT – we followed the basic functionality like creating the .dlv directory which contains the information about versions and experiments of the models.

As we can see the folder structure in the screenshots where

- It has files like
  - HEAD.txt – points the current user branch
  - Config.txt – this contains the information about the project – like name, description, author/username
- It has folders like
  - Master – default branch,
    - Status.txt – contains the information about the changed files.
    - Commit\_log.txt – contains the history of all commits/experiments.
    - Commits - this folder contains the at which stage the files are in for each experiment.
    - Cache – this folder contains the information of all the file versions. (It is created with the extensions like .1, .2, .3 ...)
  - For each user branch one folder similar to master is created

```
sindhusha55@instance-1:~/sample_project$ ls -lart
total 12
drwxrwxr-x 5 sindhusha55 sindhusha55 4096 Mar 18 15:58 Weather-Application
drwxr-xr-x 10 sindhusha55 sindhusha55 4096 Mar 19 02:32 ..
drwxrwxr-x 3 sindhusha55 sindhusha55 4096 Mar 19 03:53 .
sindhusha55@instance-1:~/sample_project$ dlv init
sindhusha55@instance-1:~/sample_project$ ls -lart
total 16
drwxrwxr-x 5 sindhusha55 sindhusha55 4096 Mar 18 15:58 Weather-Application
drwxr-xr-x 10 sindhusha55 sindhusha55 4096 Mar 19 02:32 ..
drwxrwxr-x 3 sindhusha55 sindhusha55 4096 Mar 19 03:54 .dlv
drwxrwxr-x 4 sindhusha55 sindhusha55 4096 Mar 19 03:54 .
sindhusha55@instance-1:~/sample_project$ ls -lart .dlv
total 16
drwxrwxr-x 5 sindhusha55 sindhusha55 4096 Mar 19 03:54 master
-rw-rw-r-- 1 sindhusha55 sindhusha55 6 Mar 19 03:54 HEAD.txt
-rw-rw-r-- 1 sindhusha55 sindhusha55 0 Mar 19 03:54 config.txt
drwxrwxr-x 4 sindhusha55 sindhusha55 4096 Mar 19 03:54 ..
drwxrwxr-x 3 sindhusha55 sindhusha55 4096 Mar 19 03:54 .
sindhusha55@instance-1:~/sample_project$ cat .dlv/HEAD.txt
mastersindhusha55@instance-1:~/sample_project$ ls -lart .dlv/master/
total 20
-rw-rw-r-- 1 sindhusha55 sindhusha55 0 Mar 19 03:54 status.txt
drwxrwxr-x 2 sindhusha55 sindhusha55 4096 Mar 19 03:54 stage
drwxrwxr-x 2 sindhusha55 sindhusha55 4096 Mar 19 03:54 commits
-rw-rw-r-- 1 sindhusha55 sindhusha55 0 Mar 19 03:54 commit_log.txt
drwxrwxr-x 2 sindhusha55 sindhusha55 4096 Mar 19 03:54 cache
drwxrwxr-x 3 sindhusha55 sindhusha55 4096 Mar 19 03:54 ..
drwxrwxr-x 5 sindhusha55 sindhusha55 4096 Mar 19 03:54 .
sindhusha55@instance-1:~/sample_project$
```

## 2) DLV ADD:

It adds the all locally modified files to the “stage” directory in the current branch.

```
sindhusha55@instance-1:~/sample_project$ dlv add
Please enter the directory path
sindhusha55@instance-1:~/sample_project$ dlv add -d Weather-Application/
sindhusha55@instance-1:~/sample_project$ dlv status --json
{
  "Tracked Files": [],
  "Untracked Files": [],
  "Staged Files": [
    "./Weather-Application/angular.json",
    "./Weather-Application/tsconfig.json",
    "./Weather-Application/package.json",
    "./Weather-Application/tslint.json",
    "./Weather-Application/package-lock.json",
    "./Weather-Application/e2e/protractor.conf.js",
    "./Weather-Application/e2e/tsconfig.e2e.json",
    "./Weather-Application/e2e/src/app.po.ts",
    "./Weather-Application/e2e/src/app.e2e-spec.ts",
    "./Weather-Application/src/index.html",
    "./Weather-Application/src/styles.css",
    "./Weather-Application/src/tsconfig.spec.json",
    "./Weather-Application/src/browserslist",
    "./Weather-Application/src/test.ts",
    "./Weather-Application/src/polyfills.ts",
    "./Weather-Application/src/karma.conf.js",
    "./Weather-Application/src/main.ts",
    "./Weather-Application/src/tsconfig.app.json",
    "./Weather-Application/src/favicon.ico",
    "./Weather-Application/src/tslint.json",
    "./Weather-Application/src/environments/environment.ts",
    "./Weather-Application/src/environments/environment.prod.ts",
    "./Weather-Application/src/assets/.gitkeep"
  ],
  "Modified Files": [],
  "Deleted Files": []
}
```

### 3) DLV COMMIT:

- Creates a snapshot of files and all the file versions are stored in the “cache” directory of the current branch folder.
- It updates few other files with the history like commit\_log.txt file
- It saves the current experiment with list of files and their versions.

```
sindhusha55@instance-1:~/sample_project$ dlv commit
Please specify commit message and author name
sindhusha55@instance-1:~/sample_project$ dlv commit -m "m1" -a "a1"
Committed to branch name: master with version: 1
sindhusha55@instance-1:~/sample_project$ █
```

### 4) DLV STATUS:

- It compares the files that are changed locally and with the last experiment that is saved.
- It gives the list of modified files, deleted files, tracked files, untracked files.

```
sindhusha55@instance-1:~/sample_project$ dlv status
All files in the current repository and branch: master are up-to-date
sindhusha55@instance-1:~/sample_project$ vim Weather-Application/package-lock.json
sindhusha55@instance-1:~/sample_project$ dlv status --json
{
  "Tracked Files": [
    "./Weather-Application/angular.json",
    "./Weather-Application/tsconfig.json",
    "./Weather-Application/package.json",
    "./Weather-Application/tslint.json",
    "./Weather-Application/e2e/protractor.conf.js",
    "./Weather-Application/e2e/tsconfig.e2e.json",
    "./Weather-Application/e2e/src/app.po.ts",
    "./Weather-Application/e2e/src/app.e2e-spec.ts",
    "./Weather-Application/src/index.html",
    "./Weather-Application/src/styles.css",
    "./Weather-Application/src/tsconfig.spec.json",
    "./Weather-Application/src/browserslist",
    "./Weather-Application/src/test.ts",
    "./Weather-Application/src/polyfills.ts",
    "./Weather-Application/src/karma.conf.js",
    "./Weather-Application/src/main.ts",
    "./Weather-Application/src/tsconfig.app.json",
    "./Weather-Application/src/favicon.ico",
    "./Weather-Application/src/tslint.json",
    "./Weather-Application/src/environments/environment.ts",
    "./Weather-Application/src/environments/environment.prod.ts",
    "./Weather-Application/src/assets/.gitkeep"
  ],
  "Untracked Files": [],
  "Staged Files": [],
  "Modified Files": [
    "./Weather-Application/package-lock.json"
  ],
  "Deleted Files": []
}
```

## VI. PROJECT INCREMENT 2:

### TASKS DONE:

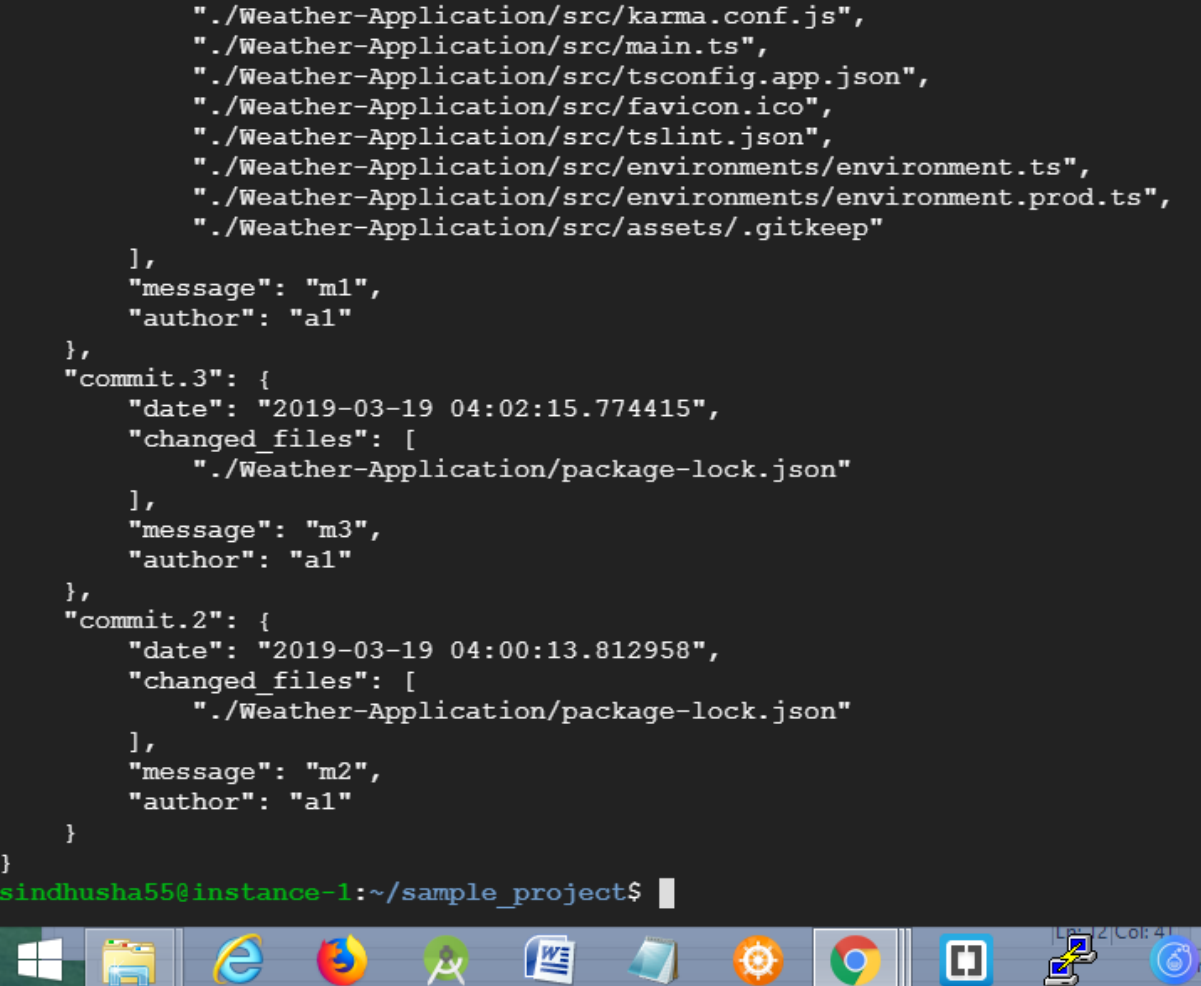
1. Creating Basic User Interface for version control system similar to GIT desktop.
2. Modifying Increment 1 commands to clear all the bugs.
3. Implemented commands {status, history, push, diff of files}
4. Implementing commands {list, desc, diff between models}

### DESIGN AND IMPLEMENTATION:

#### 1) DLV HISTORY:

- Displays the content in commit\_log.txt file present in the user's current working branch (master).

```
    "./Weather-Application/src/karma.conf.js",
    "./Weather-Application/src/main.ts",
    "./Weather-Application/src/tsconfig.app.json",
    "./Weather-Application/src/favicon.ico",
    "./Weather-Application/src/tslint.json",
    "./Weather-Application/src/environments/environment.ts",
    "./Weather-Application/src/environments/environment.prod.ts",
    "./Weather-Application/src/assets/.gitkeep"
  ],
  "message": "m1",
  "author": "a1"
},
"commit.3": {
  "date": "2019-03-19 04:02:15.774415",
  "changed_files": [
    "./Weather-Application/package-lock.json"
  ],
  "message": "m3",
  "author": "a1"
},
"commit.2": {
  "date": "2019-03-19 04:00:13.812958",
  "changed_files": [
    "./Weather-Application/package-lock.json"
  ],
  "message": "m2",
  "author": "a1"
}
}
sindhusha55@instance-1:~/sample_project$
```



#### 2) DLV DIFF:

- We are using “**difflib**” to get the diff between two files.
- If user asks for **–html** option then we are giving the diff output to **diff2html python file** – where it gives the html page with table, line numbers, added lines coloured as green, removed lines coloured as red.
- Displays the diff between two models if user specifies two models.
- Displays the diff between the current model and its previous version if only one model is specified by the user.
- Displays the diff between a file ( for current version and the previous version ) if only one file is specified by the user.
- Displays the diff between a file ( for local repository and the last snapshot version of the file )

```

siddhusha55@instance-1:~/VCS-Git-for-deep-learning/dlv_commands/lib$ python diff.py
---
+++
@@ -22,6 +22,8 @@

    cmd_parser.set_defaults(func=init)

+print(1)
+print(2)

def init(args):
@@ -39,7 +41,6 @@
    print("{repo} exists. Use '-f' to force create or remove " + dlv_dir + " directory")
    sys.exit()

- global_config.create_dlv_dir()
- global_config.create_branch(global_config.MASTER_BRANCH)
- global_config.set_branch(global_config.MASTER_BRANCH)

```

### 3) DLV LIST:

- This command gives list of projects/repositories present in the user account and the details of the repositories like( Project description, author, list of current version files).

### 4) DLV DESC:

This is similar to DLV LIST command but this command requires extra argument (model name/ project name), where it gives details about only one particular model.



## VII. PROJECT INCREMENT 3:

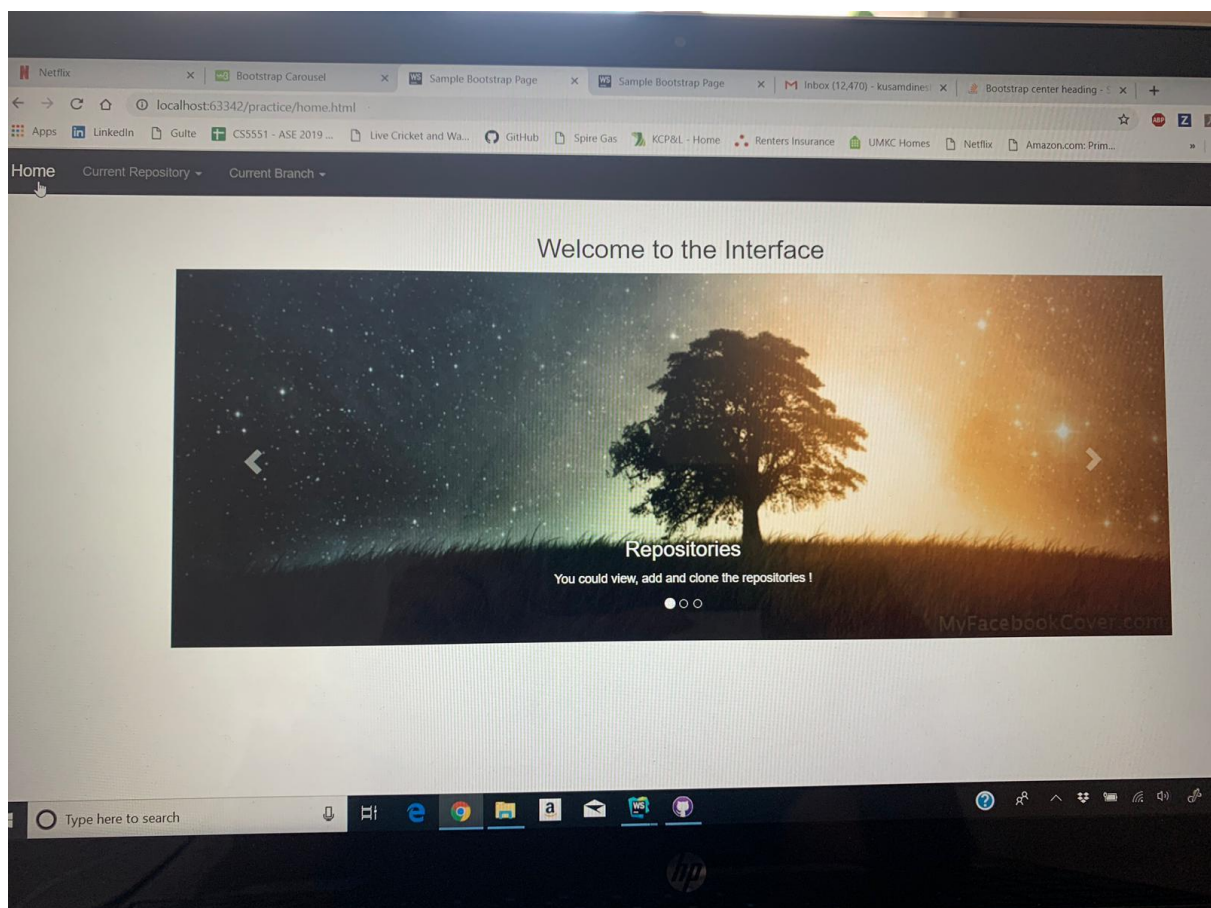
### TASKS DONE:

1. Creating Basic User Interface for Deep learning Version control system.
2. User can Register and Authenticate into DLV.
3. After Login User can upload files/repositories to the server.
4. After login User can view List of repositories.
5. On click of the repository -> It shows the status and history of the files in the repository.

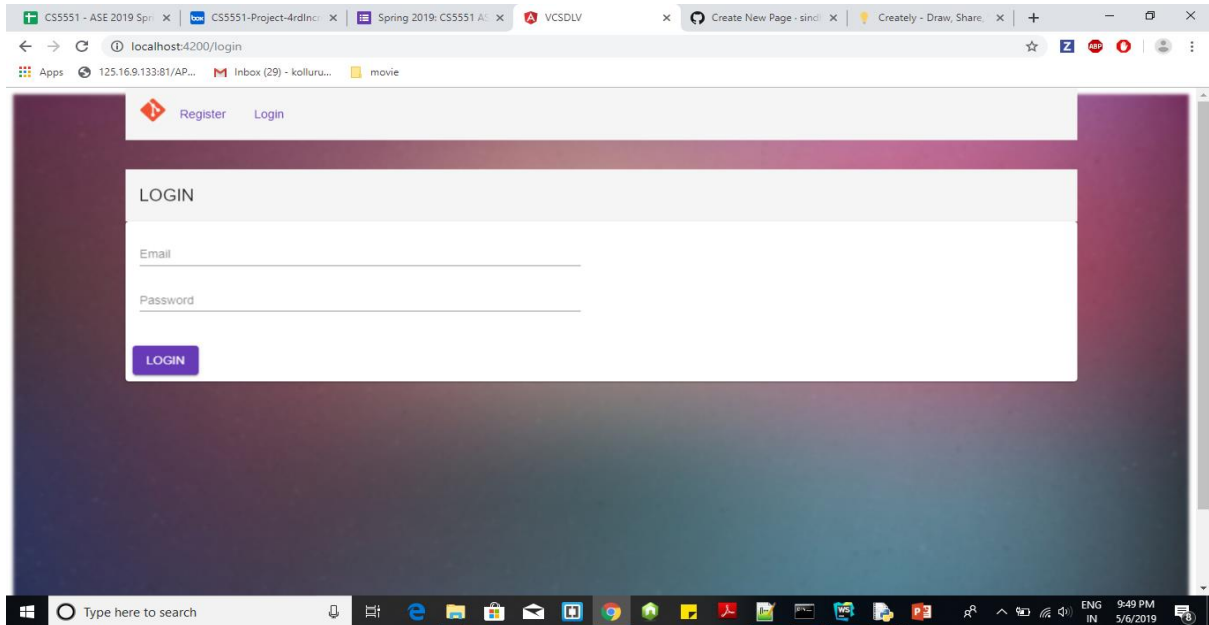
### DESIGN AND IMPLEMENTATION:

Attaching the screenshots of the User Interface:

- 1) Created a static page with the demo of the website

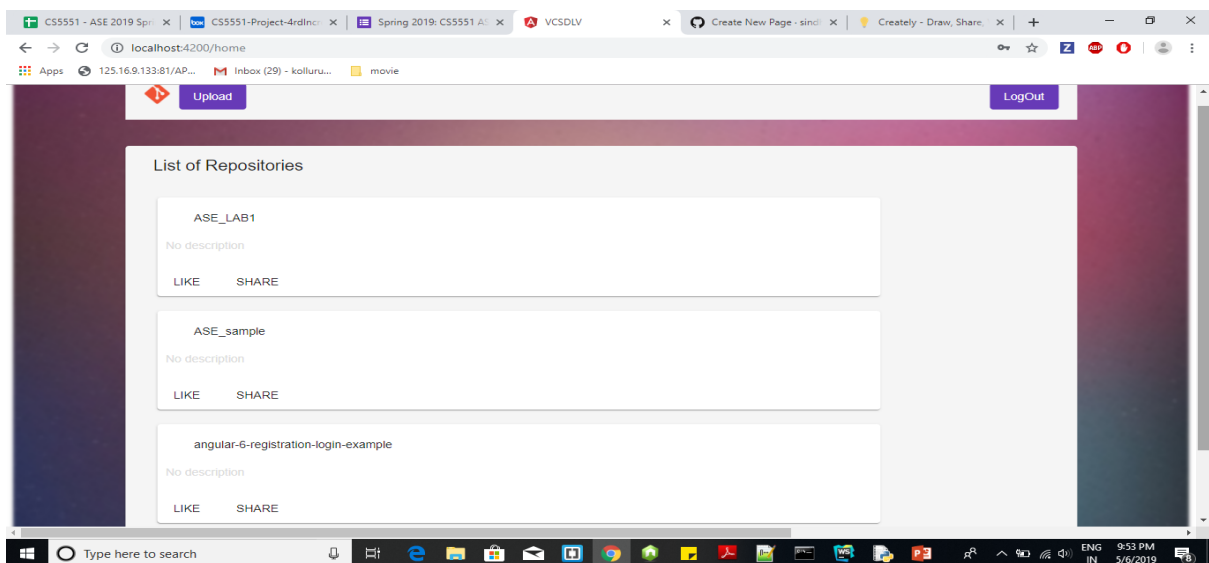


- 2) Created registration page where user can register the details and the details are stored in the MongoDB.
- 3) Created login page where user can log in to the page and the user credentials are verified by connecting to the MongoDB.



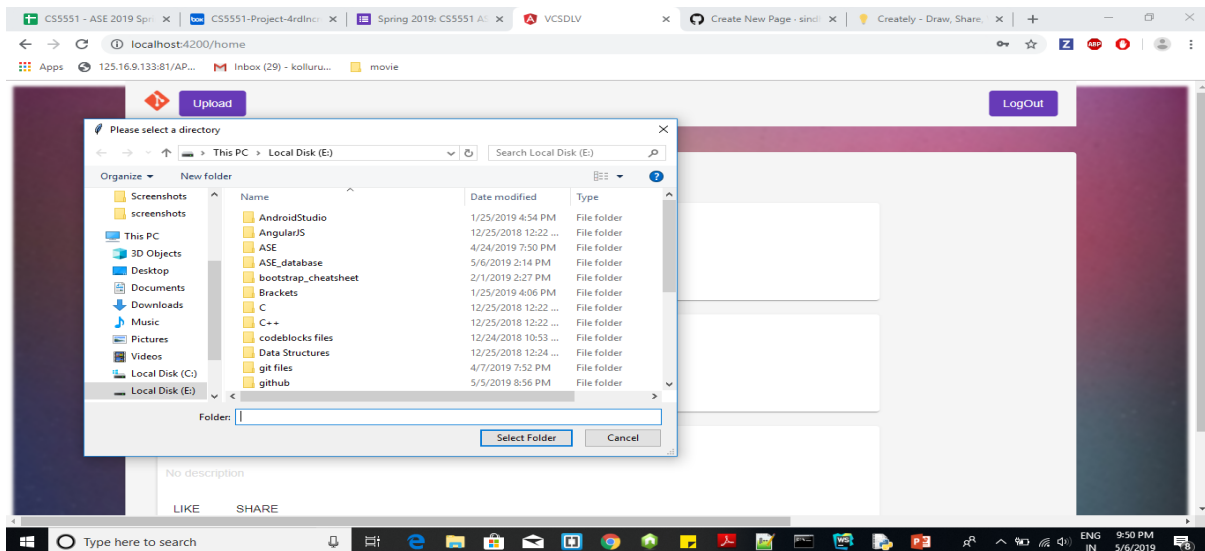
4) After logging in, User can view the list of repositories present in the user account.

- In this “**dlv list**” command is executed and the result is displayed in the UI.



5) User can upload the local project to the server. Here the server is “**Local File System**”.

- Where when user uploads the project folder – folder is saved in the separate server folder located in the local file system instead of storing in the online cloud.
- “**Dlv init**” and “**dlv commit**” commands are executed. ( this creates the initial version control system)



6) User can logout of the Application.

## VIII. PROJECT INCREMENT 4:

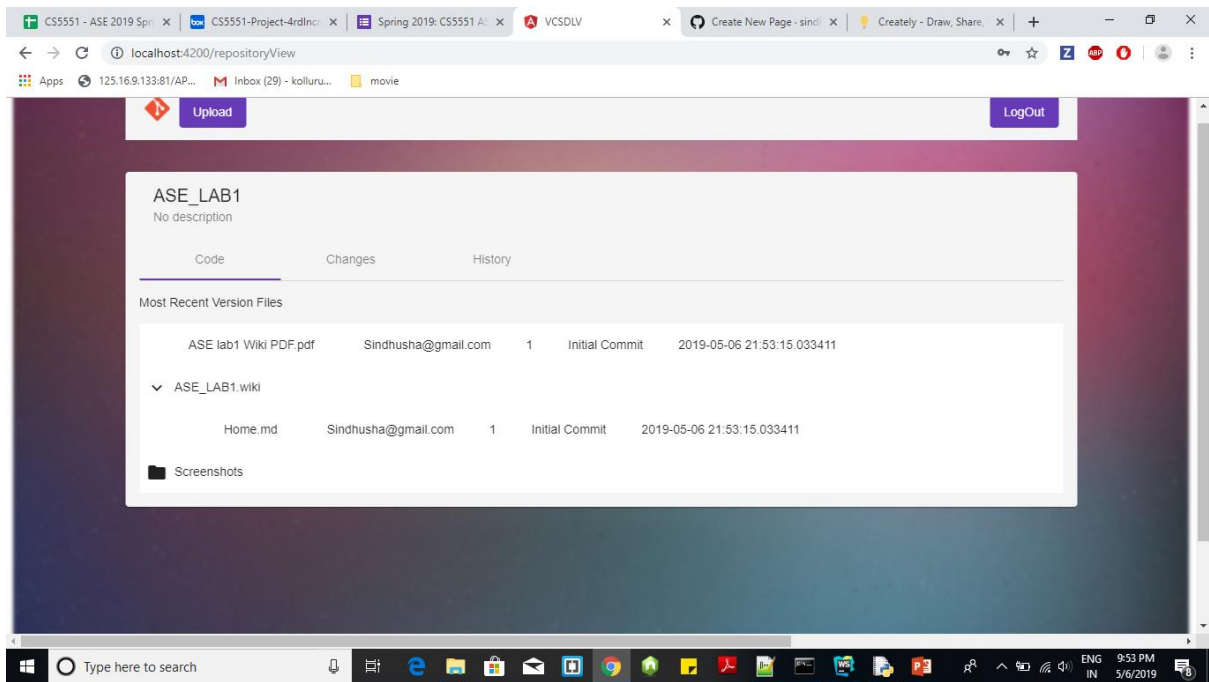
### TASKS DONE:

1. Improving the User Interface and to clear all console errors
2. After login User can view List of repositories.
3. On click of the repository -> It shows the status and history of the files in the repository.
4. Frequent polling of status GET request for every 30sec to get list of modified files.
5. Updating all details about the repository (list, status, history) after user clicks the commit.

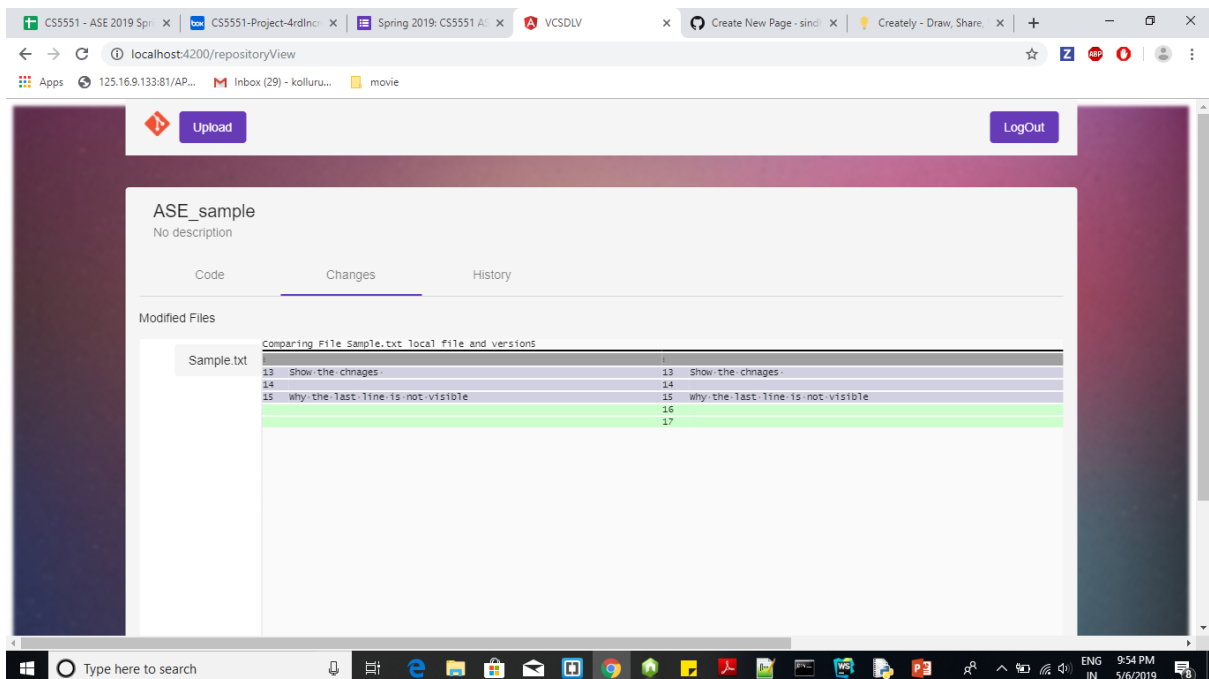
### DESIGN AND IMPLEMENTATION:

When user clicks on particular repository –

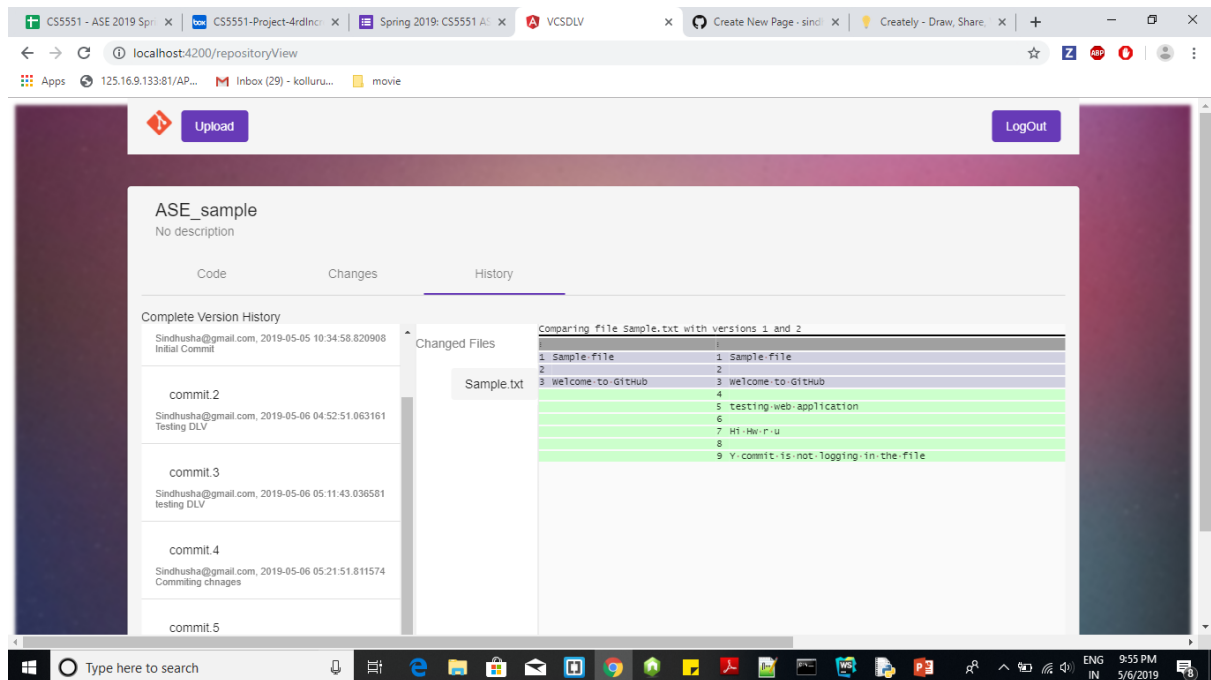
- 1) **Code TAB:** it shows the list of all files from the last experiment.
  - It shows details like file-name, folder-name, author, commit message, date
  - It executes “**dlv list**” and the result is displayed in the UI.



- 2) **History TAB:** It shows the list of all experiments in the project.
- User can view different experiments.
  - For each experiment – user can view list of changed files.
  - User can click on any file and can view the difference of files that are changed in the content.
  - It executes “**dlv history**” command and displays the result in UI.



- 3) **Changes TAB:** It shows the list of changes done to the local repository and the last experiment.
- It executes the “**dlv status**” command for every 30seconds and updates the results in the UI



## IX. WORKINGS OF PROJECT:

We have developed 3 types of servers.

- Backend command line interface (dlv commands):
  - These commands are written in python code and have created REST API using flask.
  - We have created “**bin**” directory and “**lib**” directory under the project folder dlv\_commands folder in GitHub.
  - We have added the “**bin**” directory in “**PATH environment variables**” to make it work like “**dlv init**” instead of “**python dlv init**”
    - dlv\_commands/
      - bin/
        - dlv – {python executable file}
        - folder\_selection.py – {REST API for frontend}
      - lib/
        - lib\_dlv.py – {contains the info of all commands and calls the respective function like init/add etc.,}

- global\_config.py – {contains list of global variables and functions}.
  - init.py
  - add.py
  - commit.py
  - diff.py
  - list.py
- NodeJS server for validating user – connects to MongoDB server
  - Frontend server – where the UI part and connections to backend is done through HTTP request.

## **X. FUTURE WORK:**

- 1) Creating executable file from the python dlv commands for command line interface.
- 2) Implementing search command (across users –complete database)
- 3) Reverting back to the previous changes (complete folder just like tags and branches in GIT).

## **XI. REFERENCES:**

[1] Hui Miao, Ang Li, Larry S. Davis, Amol Deshpande, "ModelHub: Deep Learning Lifecycle Management" 2017. <https://par.nsf.gov/servlets/purl/10041785> [2] Hui Miao, Ang Li, Larry S. Davis, Amol Deshpande, "Towards Unified Data and Lifecycle Management for Deep Learning" 2016 <https://arxiv.org/pdf/1611.06224.pdf>

[2] GitHub Internals - <https://github.com/pluralsight/git-internals-pdf>

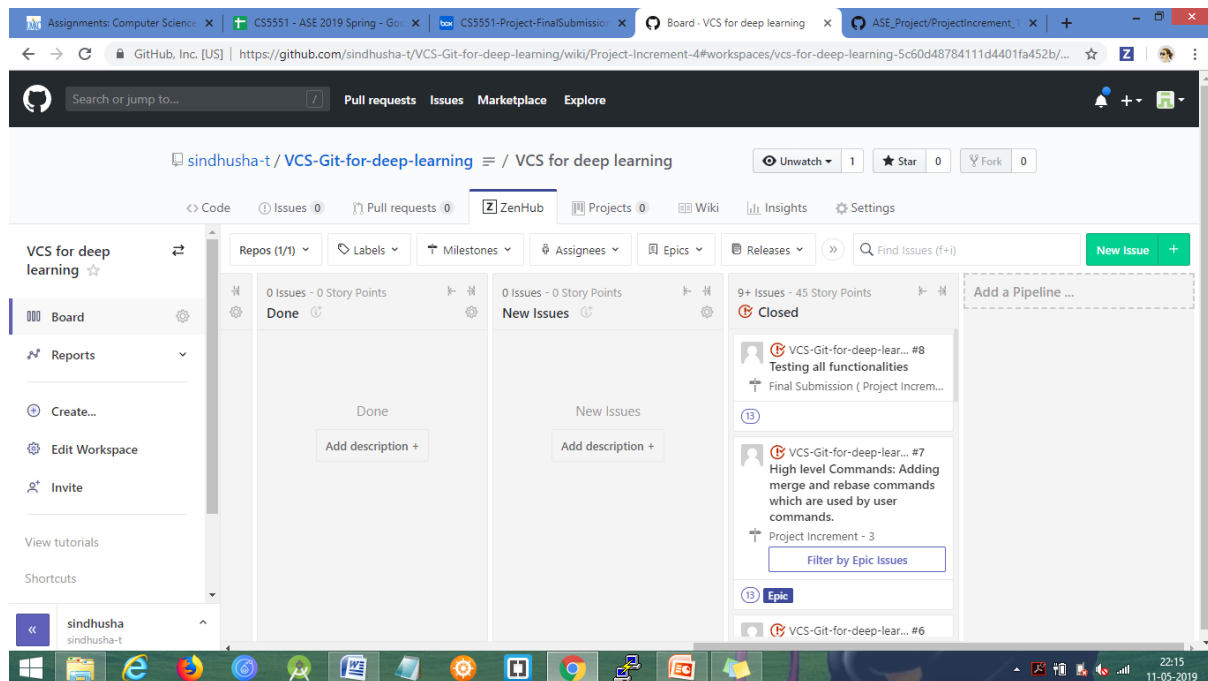
[3] Complete tutorial on Git by GitHub - <https://git-scm.com/book/en/v2>

## **XII. PROJECT DEPLOYMENT:**

- 1) We deployed the backend server for validating the user to heroku.  
URL: <https://aseprojectapp.herokuapp.com/user>

2) As we are using local file system and tkinter library we were unable to deploy the dlv commands written in python to heroku.

3) As Web application has dependency with python commands – we are not deploying the angular code to heroku.



## XIII. PROJECT MANAGEMENT:

**Equal Individual Contribution – 25% each.**

- Every member of the project contributed solely for the project.
- We have discussed at every stage about the design, implementation, and how to divide the work.
- We divided the work and followed the Agile process to complete the project.
- We spend some time together to clear each other doubts and integrate each one module into a single project.

### Project Management Report:

#### 1) Individual Contribution:

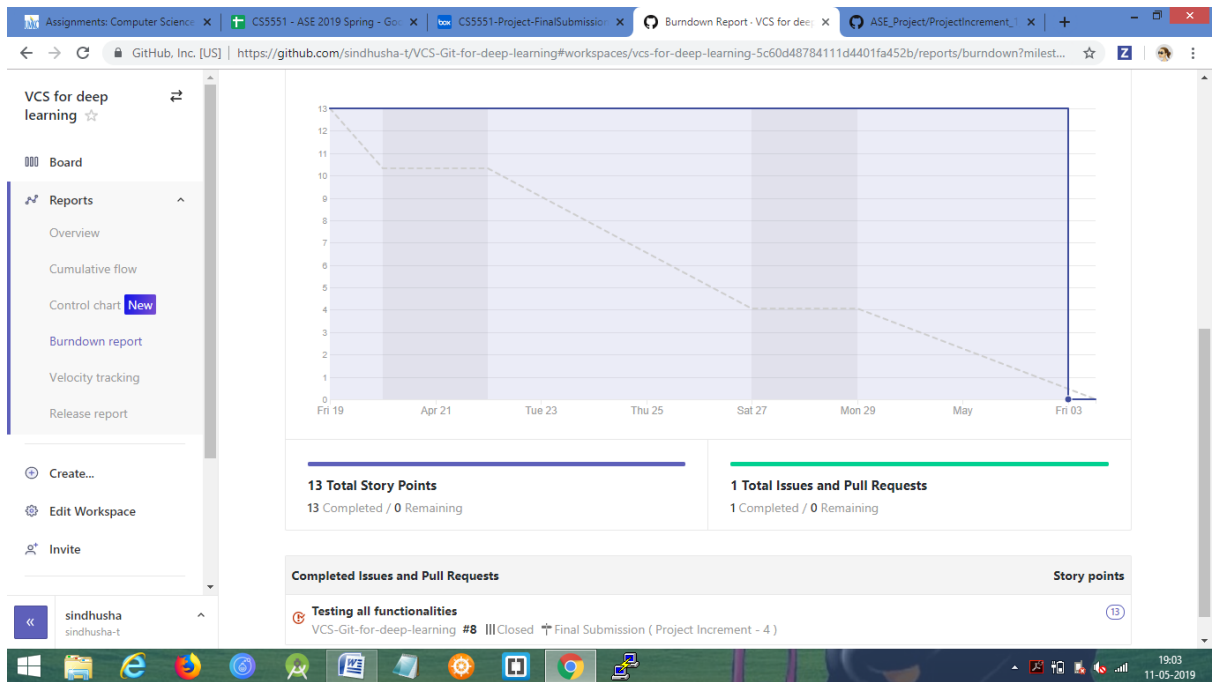
- **Increment 1:**
  - In first increment we have gone over the existing git internals and discussed to clarify each other doubts.

- We had discussions on the proposed design.
  - **Sindhusha:** Implemented dlvs status command and the integration of all commands into the dlvs executable python file.
  - **Pradeepika:** Implemented dlvs add command
  - **Sravan:** Implemented dlvs init command
  - **Dinesh:** Implemented dlvs commit command
- **Increment 2:**
    - We have modified the proposed design and re implemented all the commands.
    - **Sindhusha:** Implemented dlvs list and diff commands
    - **Pradeepika:** Implemented dlvs history and desc commands
    - **Sravan:** Implemented the UI part (with the demo page and the changes/history pages in bootstrap).
    - **Dinesh:** Implemented the UI part (with the demo page and the repositories view page in bootstrap).
- **Increment 3:**
    - **Sindhusha:** Implemented the integration part of python and UI – where uploading the repositories, viewing list of repositories.
    - **Pradeepika:** Implemented login and registration UI part
    - **Sravan:** Implemented the backend NodeJS part connection to the MongoDB
    - **Dinesh:** Implemented the UI part home page.
- **Increment 4:**
    - **Sindhusha:** Implemented the integration part of python and UI – where continuously polling the status command, showing the modified files, diff content pretty print, history of the files/versions/experiments, current version files in the project.
    - **Pradeepika:** Implemented UI part for current version of files.
    - **Sravan:** Implemented UI part for the changes tab in the home page.
    - **Dinesh:** Implemented UI part for the history tab in the home page.

## 2) The Screenshots of the project planning and task management:

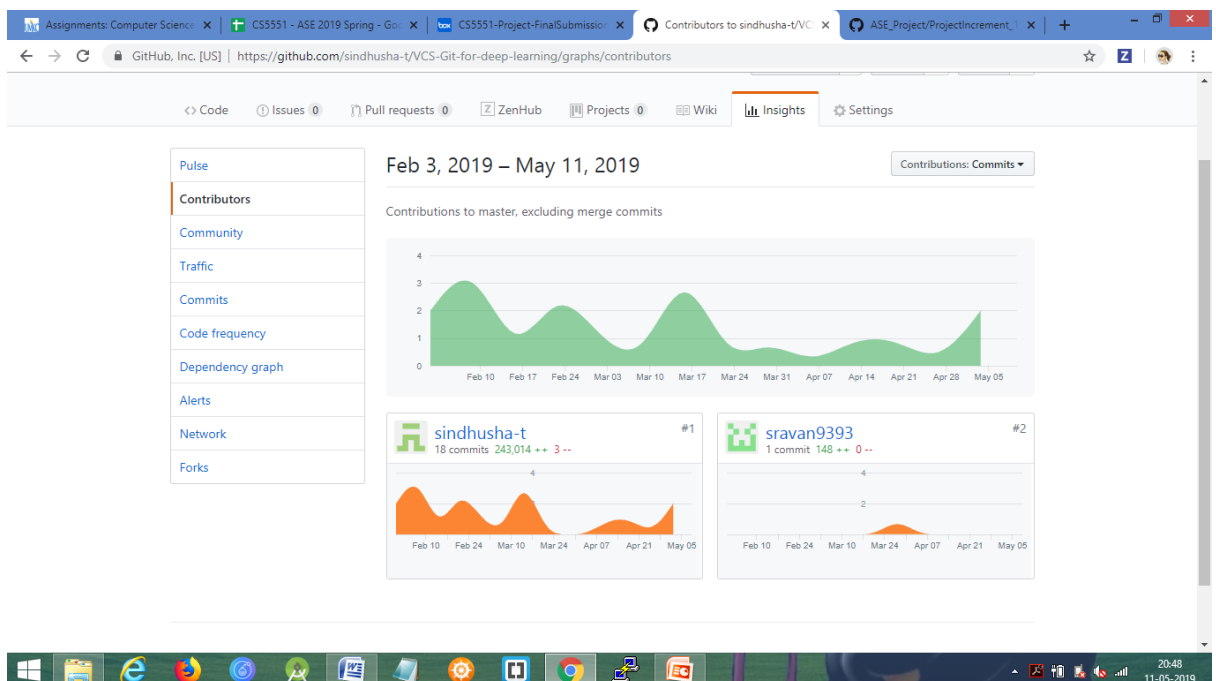
**Burn down chart:**





## CONTRIBUTION REPORT:

Each one of us have done changes to the branches individually and then merged the changes with the master branch.



## Final Project Evaluation:

- We have learnt the internal workings of the GIT functionality. How it handles the version control system.

- We have learnt the advantages and disadvantages of different source code management systems like CVS, GIT, mercury, subversion etc.,
- We have understood which version control system best suits for which system.
- Mainly GIT is built for the usage of open source projects and as expected it has become most popular for open source projects.
- Apart from this – we have initially chosen the exact GIT functionality, where we have faced few issues while developing. We have learnt throughout the journey.
- So keeping the deadlines in mind – we have modified few design parts from the GIT Internals to make the work easier.
- Agile process was really helpful for the project. We were able to equally divide the work among ourselves and do the need before deadlines and integrate all the work.
- We have implemented most of the features and few features were included in the future work.
- We have met and discussed many times for each increment for clearing each other doubts.

## **ACKNOWLEDGEMENT STATEMENT**

The work has been completed under the guidance of

Dr. Yugi Lee,  
Rajaram Anantharaman,  
and TAs ( Sirisha Rella, Bhargavi Nadendla)

in CS5551 Advanced Software Engineering,  
University of Missouri - Kansas City), spring 2019.