GitStud: A Web-based Application for Sharing Project Files

Mohammad Bsoul, Emad E. Abdallah, Qais Saif, Ibrahim Albarghouthi, Mohammed Albeddawi, Essam Qaddurah
Faculty of Prince Al-Hussein Bin Abdullah II for Information Technology
The Hashemite University email:mbsoul@hu.edu.jo
Zarqa, Jordan

***Abstract*—The aim of this paper is implementing a web-based application for facilitating the communication of the members of the same project (assignment) group. Most of the group work needs to be organized and shared between teammates to enable team working and collaboration, which makes it easier and faster to accomplish tasks. Our contribution to this field is a web-based application that grants access to a repository with complete history of all files regarding a specific project. This web-based application named GitStud is expected to make it easier for students who work in groups to organize their work and keep track of all files that are being committed up-to-date. GitStud is based on sharing directories, which makes storing files from local to global. We used Git version control software to implement our web-based application. Git version control software is considered an open source, distributed version control system that can handle projects of different sizes.**

Keywords–Git; web-based application; Students; Instructors; Projects.

I. INTRODUCTION

The students who work on the same project might face difficulties in communicating with each other. Team work needs cooperation between teammates, and this could be done better by sharing the project files with other teammates. The process of sharing files requires a shared repository which provides everyone with an access to the project files.

The files related to a project should be viewed by all team members to see and edit as needed. Git version control software [1] allows the user to track the history of a collection of files and includes the functionality to revert a file back to its previous version. Each version captures a snapshot of the file at a certain time. The collection of files belonging to the same project files and their complete history are stored in a repository. This way, losing files or any conflict in the development process will be avoided. There is a number of hosting services Git repositories such as Assembla, Beanstalk, Bitbucket, CloudForge, Codebase, Fog Creek Kiln, GitHub, GitLab, Planio, Perforce, RhodeCode, and Unfuddle [2].Git development began in 2005. Git was originally designed as a low-level version control system engine on top of which others could write front ends. Then, the core Git project has become a complete version control software that can be used directly. However, the existing version control software is not dedicated for educational purposes where the intended users are students and instructors. This kind of software is needed to facilitate the communication between students of the same group and between students and instructors.

In this paper, we implemented a web-based application named GitStud and which is based on Git. GitStud provides a web-based graphical interface, which is much easier to work with than Git which is strictly a command-line tool. GitStud is expected to help improve collaboration and eases the communication process between the project members.

The rest of this paper is structured as follows. Related works are presented in Section II. Section III describes the new web-based application. In Section IV, we show some screen results from the web-based application. Finally, Section V concludes the paper and describes future work.

II. RELATED WORKS

In this section, we will discuss the differences between our implemented GitStud and other version control software (including Git that we used in our implemented GitStud).

Bazaar [3] can be used by either a developer that is working on many branches which have local content or by groups sharing work through a network. Bazaar is free software written in Python programming language.

Darcs [4] is a distributed version control system that has many features such as the ability to select which changes to accept, and interaction with local or remote repositories.

Mercurial [5] is a cross-platform and distributed revision control tool that can be used by software developers and It is written in Python programming language. Mercurial is a command line program.

Revision Control System [6] is a software that automates the storing, retrieval, logging, identification, and merging of changes. The Revision Control System is useful for text that is modified frequently.

Apache Subversion [7] is a software for maintaining current and historical versions of files.

StarTeam [8] is employed in software development, especially when a project involves many teams in various places.

Code Co-op [9]is a system that employs peer-to-peer architecture to share projects and to control modifications to files. Code Co-op replicates its database on every device in the project.

PTC Integrity [10] uses a client/server model and allows software developers to track their work.

Git has many features over other version control software. One of its features is its branching model. In this model, it is allowed to have multiple local branches that are entirely independent of each other. The creation, merging, and deletion of these lines of development only needs seconds. Additionally, Git treats the stored data as a stream of snapshots. As a result, when a project is committed, or saved in Git, it takes a snapshot of what all the files look like at that moment and adds a reference to the snapshot. If there is no change on the file, Git does not store the file again, but only adds a link to the previous identical file.

None of the above version control software is dedicated for educational purposes. Therefore, there is a need for a software that facilitates the communication between students in the same project. Additionally, this software is needed to facilitate the communication between students and instructors who create assignments for these students.

The aim of our web-based application called GitStud is implementing a web-based application using Git and deploy it to ease the communication between the project members.

III. GITSTUD WEB-BASED APPLICATION

This paper is about implementing a web-based application using Git. Its target is making the communication process easier between the students of the same project group (team-mates). GitStud provides a user interface that is used by the group members to work on the files of the project (assignment). The provided user interface includes all the functionalities needed to work on the project posted by the instructor on GitStud.

GitStud offers a time saving solution for students to overcome team-working problems. GitStud supports the communication between students of the same group and between students and instructors by providing a global repository that is based on Git service.

GitStud requirements are:

- Web browser.
- Latest version of Java.
- Git software.
- Internet connection.

Figure 1 shows the workflow diagram of GitStud. One of the student (named Super Student) in the project group has first to login to his account in order to create a repository for the project. This created repository will contain the files that belong to the project. Then, all the students in this project group can pull the files in this repository in order to work on them and then push them back into the repository after they finish. Upon completion of the project, their work can be submitted to the instructor who posted the project (assignment) for evaluation.

Figure 2 shows the use case diagram, and Tables I to IX show detailed descriptions of use cases.

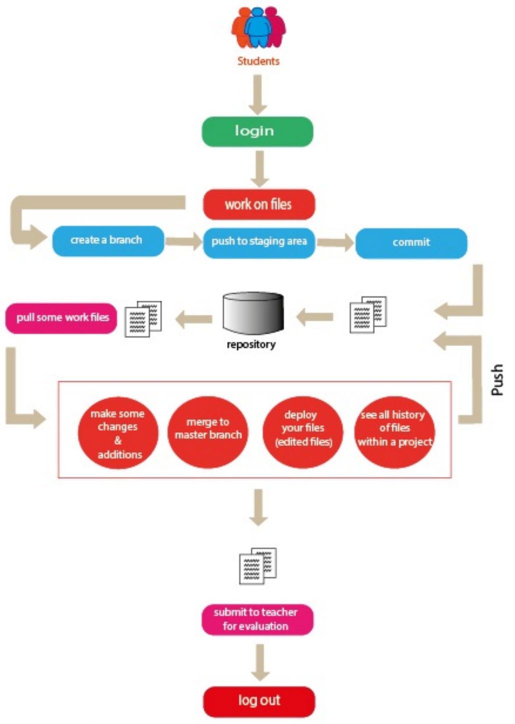

Students
login
work on files
create a branch
push to staging area
commit
pull some work files
repository
Push
make some changes & additions
merge to master branch
deploy your files (edited files)
see all history of files within a project
submit to teacher for evaluation
log out

Figure 1. Workflow diagram.

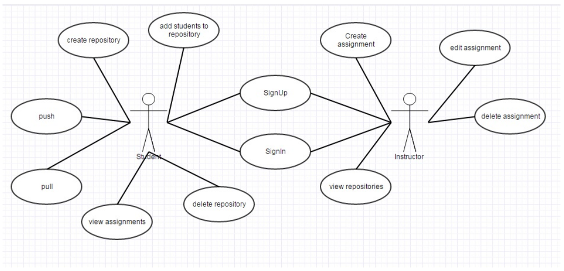


Figure 2. Use case diagram.

TABLE I. SIGN UP USE CASE.

Name	Sign up.
Brief description	This use case is the basic step to enter the system. It ensures security and authentication for users.
Actors	Students, instructors.
Pre-conditions	The system should be up and running and the sign up form must be opened.
Main flow of events	A user opens GitStud and clicks the sign up button to open the sign up form. Then, the user chooses if he is a student or instructor. Next, the user provides the needed information such as the user name, e-mail, password and student ID/instructor ID and click on sign up button to confirm his information.
Alternative flow of events	A user opens GitStud and clicks the sign up button to open the sign up form. Then, the user provides the needed information. The information that the user entered appears that it already exists in the system such as the user name, e-mail and the student ID/instructor ID. If username, e-mail, or student ID/instructor ID exists, the user is prompted that he has an existing account.
Post-conditions	If the sign up process went successfully the system shows a message to the user to inform him that the sign up process was successful. Otherwise, the user will be informed to enter the required missing information or to correct the incorrect information.

TABLE II. SIGN IN USE CASE.

Name	Sign in.
Brief description	This step allows the user to access the system in order to use the features and functions of GitSud.
Actors	Students, instructors.
Pre-conditions	The user must have an existing account on GitStud to be able to access the system.
Main flow of events	A user enters his username and password in the sign in form. After successful sign in, the user will have access to GitStud's dashboard.
Alternative flow of events	A user enters incorrect username and password in the sign in form. In this case, the system notifies the user that his user name or password is incorrect.
Post-conditions	After signing in, GitStud's dashboard will appear and the user will have access to functions of the system.

TABLE III. CREATE REPOSITORY USE CASE.

Name	Create Repository.
Brief description	The creation of the repository allows students to have a shared data store to add their work files to.
Actors	Students.
Pre-conditions	Student must have an account to see this function.
Main flow of events	A student sign in. Then, he student chooses the create repository feature. Next, the student names the new repository. Then, the student sees the available courses. Next, the student chooses one of the courses in order to see the assignments for this course. If the student chooses an assignment, an empty repository is created for it. This student will become the super student for this repository and will have full control over it.
Alternative flow of events	A student chooses to create a new repository. This student has another repository with the same name. The student is prompted to rename the new repository. The student has to rename the new repository.
Post-conditions	The new empty repository is created and ready for use.

TABLE IV. PULL USE CASE.

Name	Pull.
Brief description	A pull request updates current local directory to up-to-date files in the repository.
Actors	Students.
Pre-conditions	A student must share a repository and the repository has work files to be pulled. The student must first clone the repository to a local directory. The student ensures that the local directory is not up-to-date.
Main flow of events	A student clicks on the pull button from the desktop application.
Alternative flow of events	A student tries to pull some files before cloning the repository. The student is asked to clone the repository first to be able to pull latest files.
Post-conditions	New work files are added to his local directory.

TABLE V. PUSH USE CASE.

Name	Push.
Brief description	The push feature adds modified files to the repository.
Actors	Students.
Pre-conditions	A student must share a repository. The student must first clone the repository to a local directory. Next, the student ensures that the local directory is modified. Then, the student has to commit changes.
Main flow of events	A student clicks the push button from the desktop application.
Alternative flow of events	A student tries to push some files before cloning the repository. In this case, the student is asked to clone repository first. The student must commit before push operation.
Post-conditions	The files are added to the repository with information about the commit activity.

TABLE VI. COMMIT USE CASE.

Name	Commit.
Brief description	The commit adds files that will be pushed to a repository.
Actors	Students.
Pre-conditions	A student has modified files and wants to add them to the repository.
Main flow of events	A student works on files on the local directory. Then, the student selects the files that are ready for the push process and adds a message with the commit.
Alternative flow of events	The student did not select the files to be committed. In this case, the student is asked to select the files first.
Post-conditions	The files are added to the staging area and are ready to be pushed.

TABLE VII. VIEW HISTORY USE CASE.

Name	View history.
Brief description	This feature allows user to trace full history of commits.
Actors	Students, instructors.
Pre-conditions	The user must have an access to the repository.
Main flow of events	The User selects the repository that he wants to view its history. The user clicks on view history.
Alternative flow of events	——————————
Post-conditions	The User can move between commits.

TABLE VIII. CREATE ASSIGNMENT (PROJECT) USE CASE.

Name	Create assignment.
Brief description	An instructor adds an assignment for a specific course so students can work on.
Actors	Instructors.
Pre-conditions	The instructor must have added at least one course.
Main flow of events	The instructor chooses a course then fills the assignment form. Next, he clicks on the upload button to upload the required assignment.
Alternative flow of events	The instructor did not fill the required fields. The instructor is notified to fill the empty fields.
Post-conditions	The students in the course are informed about the assignment.

TABLE IX. EDIT ASSIGNMENT (PROJECT) USE CASE.

Name	Edit assignment.
Brief description	An instructor modifies an assignment.
Actors	Instructors.
Pre-conditions	The instructor must have added at least one assignment to the course.
Main flow of events	The instructor chooses a certain assignment. Then, the instructor clicks on edit assignment button. Next, the instructor modifies the assignment and clicks on save.
Alternative flow of events	The instructor modifies a field by leaving it empty. The instructor is notified that he left a field empty.
Post-conditions	The students of the assignment are notified of the change made to the assignment.

Figure 3 shows the class diagram and the variables and methods of each entity (class). Super student entity represents the student (one of the project members) who is responsible for repository creation, deletion, and management.

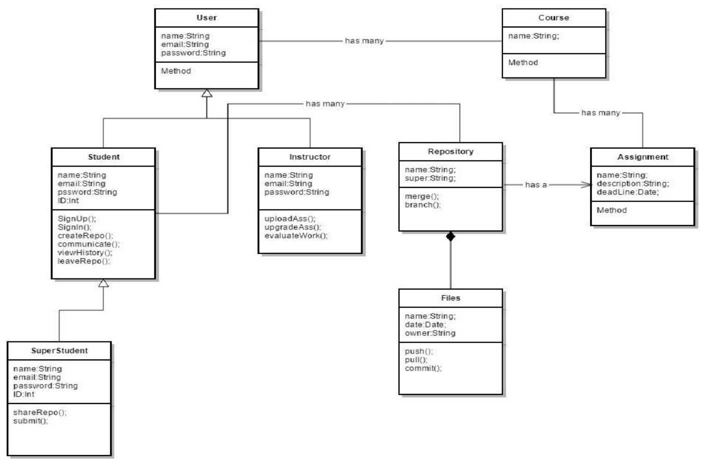
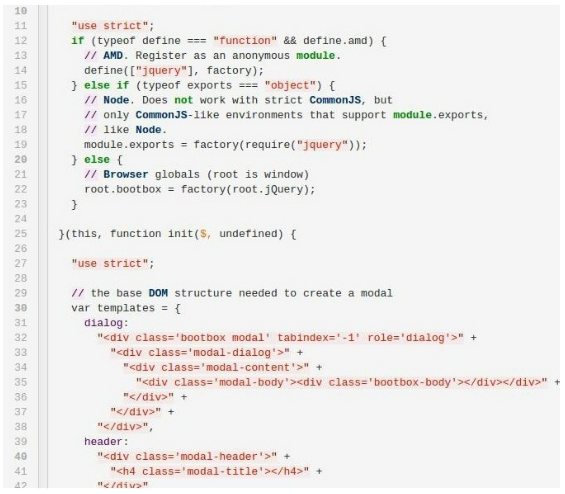


Figure 3. Class diagram.

IV. RESULT SCREENS

In this section, we show some screen results from our implemented web-based application.

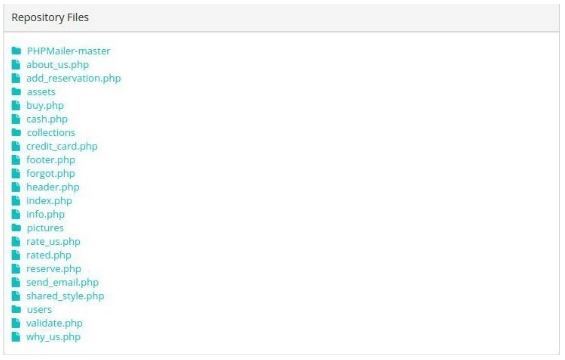
Figure 4 shows the content of one of the files belonging to the repository of an assignment (project) group.



```
    "use strict";
    if (typeof define === "function" && define.amd) {
      // AMD. Register as an anonymous module.
      define(["jquery"], factory);
    } else if (typeof exports === "object") {
      // Node. Does not work with strict CommonJS, but
      // only CommonJS-like environments that support module.exports,
      // like Node.
      module.exports = factory(require("jquery"));
    } else {
      // Browser globals (root is window)
      root.bootbox = factory(root.jQuery);
    }

  }(this, function init($, undefined) {

    "use strict";

    // the base DOM structure needed to create a modal
    var templates = {
      dialog:
        "<div class='bootbox modal' tabindex='-1' role='dialog'>" +
          "<div class='modal-dialog'>" +
            "<div class='modal-content'>" +
              "<div class='modal-body'><div class='bootbox-body'></div></div>" +
            "</div>" +
          "</div>" +
        "</div>",
      header:
        "<div class='modal-header'>" +
          "<h4 class='modal-title'></h4>" +
        "</div>"
```

Figure 4. In file screen.

In Figure 5, it can be seen the files in one of the assignment repositories.



Repository Files

- PHPMailer-master
- about_us.php
- add_reservation.php
- assets
- buy.php
- cash.php
- collections
- credit_card.php
- footer.php
- forgot.php
- header.php
- index.php
- info.php
- pictures
- rate_us.php
- rated.php
- reserve.php
- send_email.php
- shared_style.php
- users
- validate.php
- why_us.php

Figure 5. Repository files screen.

Figure 6 shows the screen that appears to the instructor when he wants to create a new assignment.

In Figure 7, it is shown the assignments created by one of the instructors.

V. Conclusion

In this paper, we have implemented a web-based application named GitStud using Git.This web-based application has been designed to make it easier for students who work in groups to organize their work and keep track of all files that are being committed up-to-date.

GitStud allows students to track the history of a collection of files and includes the functionality to revert the collection of files back to the previous version. In this way, losing files or any conflict in work will be avoided. It is also expected to help improve collaboration and ease the communication process between the students of the same project group.

In future work, we plan to add e-learning feature where the students can watch the lectures posted by instructors. Additionally, we plan to support more languages. Finally, we plan to have user testing for GitStud to prove that it makes it easier for students to work collaboratively, enhance communication between them in team-work situations and work together on files.

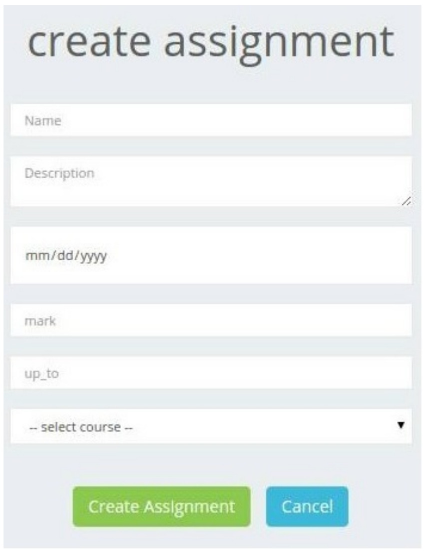
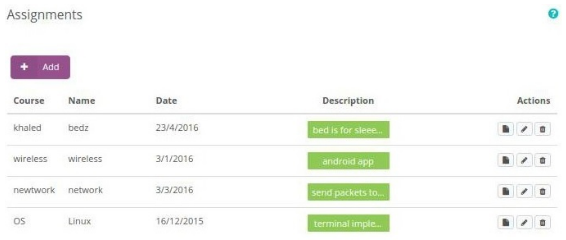


Figure 6. Create assignment screen.



Assignments

+ Add

Course	Name	Date	Description	Actions
khaled	bedz	23/4/2016	bed is for sleee...	
wireless	wireless	3/1/2016	android app	
newtwork	network	3/3/2016	send packets to...	
OS	Linux	16/12/2015	terminal imple...	

Figure 7. Instructor assignments screen.

References

[1] https://git-scm.com, [retrieved: July, 2016].

[2] T. Gunther, “12 git hosting services compared,” http://www.git-tower.com/blog/git-hosting-services-compared/, 2014, [retrieved: July, 2016].

[3] http://bazaar.canonical.com/en/, [retrieved: July, 2016].

[4] http://darcs.net/, [retrieved: July, 2016].

[5] https://www.mercurial-scm.org/, [retrieved: July, 2016].

[6] http://www.gnu.org/software/rcs/rcs.html, [retrieved: July, 2016].

[7] https://subversion.apache.org/, [retrieved: July, 2016].

[8] http://www.borland.com/en-GB/Products/Change-Management/StarTeam, [retrieved: July, 2016].

[9] http://www.relisoft.com/co_op/index.htm, [retrieved: July, 2016].

[10] http://www.ptc.com/application-lifecycle-management/integrity, [retrieved: July, 2016].