Collaborative Project Developer

Undertaking

We declare that work presented in this project titled "Collaborative Project Developer" , submitted to Prof. M.M Gore , Computer Science and Engineering , Motilal Nehru National Institute of Technology is an original work . We have not plagiarized or submitted the same work for the award of any other degree.

November, 2015, Allahabad

Saquib Aftab (20124027)

Sandeep Kumar (20121059)

Ravi Ratan (20124026)

Ravi Shankar (20124025)

Acknowledgement

We would take this opportunity to thank our teacher, Prof. M.M.Gore for providing us with such an exciting project to work on. His constant support and motivation helped us to do our best on the project. We thank him for all his help and excellent guidance upon us to help us achieve the result that we have on our project.

Table of Contents

Chapter 1. Introduction

Collaborative Project Developer is a very simple tool which enable us to collaborate on a project and maintain version. In a project it is very important to maintain versions and commit history, but the VCS which are available requires getting used to it. So, we started developing `Collaborative Project Developer`, which is very simple to use even for the first timer.

1.1. Motivation

We wanted to develop an application, which can help people start collaborating on project development without having to go learn all the technicalities of version control systems such as git and perforce. So, we started working on this simple versioning and collaborative tool.

1.2. Algorithm

In transaction processing, databases, and computer networking, the two-phase commit protocol (2PC) is a type of atomic commitment protocol (ACP). It is a distributed algorithm that coordinates all the processes that participate in a distributed atomic transaction on whether to commit or abort (roll back) the transaction (it is a specialized type of consensus protocol). The protocol achieves its goal even in many cases of temporary system failure (involving either process, network node, communication, etc. failures).

1.3. Platform

We are using node.js as our platform for developing this project. Alongwith node.js we are using some libraries to make development process faster.

1.3.1. Node.js

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world. .An Example of node.js server

```
var http = require('http');
```

```
//Lets define a port we want to listen to
const PORT=8080;

//We need a function which handles requests and send response
function handleRequest(request, response){
    response.end('It Works!! Path Hit: ' + request.url);
}

//Create a server
var server = http.createServer(handleRequest);

//Lets start our server
server.listen(PORT, function(){
    //Callback triggered when server is successfully listening. Hurray!
    console.log("Server listening on: http://localhost:%s", PORT);
});
```

1

1.3.2. Express

Express.js is a Node.js web application server framework, designed for building single-page, multi-page, and hybrid web applications.[1] It is the de facto standard server framework for node.js.[2] The original author, TJ Holowaychuck, described it as a Sinatra-inspired server,[3] meaning that it is relatively minimal with many features available as plugins.

1.3.3. Mongoose

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

An Example of mongoose

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

var Cat = mongoose.model('Cat', { name: String });

var kitty = new Cat({ name: 'Zildjian' });
kitty.save(function (err) {
```

[1] A simple node server

```
  if (err) // ...
  console.log('meow');
});
```

[2]

1.3.4. Passport

Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, and more.

1.3.5. Codemirror

Codemirror is an open source library used for implementing code editor in web pages. We have used this plug-in for implementing the code editor that is displayed on our web-app.

[2] Mongoose example

Chapter 2. Structure of the App

The structure of our app is inspired from MVC design philosophy. In MVC(Model View Controller), we separate what we have to show(views) from the main logic(controller). Similarly we also separate the models of underlying structure(models).

In our main app, we have the following directories/files.

2.1. Views

In this directory, we write all the files that we have to display. For example, `index.html`, `login.html` and `signup.html`.

2.2. Routes

In this directory, we write all the server routes. For example, to handle the request for approving the code by a particular team member, we wrote the following code in `index.js` in routes directory:

```
router.get('/accept/:id', function(req, res, next) {
  Code.findOne({_id: req.param("id")},function(err, doc){
    var idxt = doc.approval.indexOf(req.user.local.email);
    if(doc.approval[idxt] != req.user.local.email){
      doc.approval.push(req.user.local.email);
      console.log("index in approval is"+idxt);
    }
    var idx = doc.rejection.indexOf(req.user.local.email);
    if(doc.rejection[idx]==req.user.local.email){
      doc.rejection.splice(idx, 1);
      console.log("index in rejection is"+idx);
    }
    res.send("OK");
    console.log("Updated Approval and Rejection List are:");
    console.log(doc.approval);
    console.log(doc.rejection);
    updateCommit(doc);
    console.log(doc.committed);
    doc.save();
  });
});
```

2.3. node_modules

This directory contains all the node libraries that we use in our app.

2.4. public

This directory houses static files(CSS or front end libraries) available.

2.5. bin

This directory houses the `www` script that contains environment configurations for our app.

2.6. app.js

This is our main server. This file is responsible for importing all the libraries, configuring the environment and starting the server.

Chapter 3. Features of the app

3.1. Authentication

This app provides the feature for the authentic members to log into the app. This prevents the unauthorised users from making any changes or corrupting the files. For Authentication purpose, we use `passport.js` library.

3.2. Online Code Editor

Using `codemirror` library as a starting point, we were able to implement an online code editor for our app. This is a full-featured online-editor.

3.2.1. Syntax highlighting

The code-editor has the capability to highlight syntax of the code.

3.2.2. Auto-indent

This code editor also provides facility for auto-indenting codes, while changing lines.

3.2.3. Line Numbers

Line numbers also appear on our code-editor.

3.3. History of commits

Our app provides, commit history to be able to see previous commits and the person who made that commit.

3.4. Rollback to previous version

We also provide feature to roll-back to the previous version of the commit. If something goes wrong in the latest version of the project, we have the feature to roll-back to previous stable version of the project.

Glossary

List of Glossaries are:

node.js	Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i.
Express	App framework used in our app.
Mongoose	middleware used in our app to access database.
codemirror	Library used for implementing code editor in web-sites.

Index