GitGoing Code Review

# Software Design Document

Names: Travis Lamb, Christina Nguyen, Michael Bloomquist, Sarah Ramanazi

Date: (11/09/2020)

**TABLE OF CONTENTS**

1. [INTRODUCTION 2](#_bookmark0)
   1. [Purpose 2](#_bookmark0)
   2. [Scope 2](#_bookmark0)
   3. Definitions, Acronyms, and Abbreviations
   4. [References 2](#_bookmark0)
2. [SYSTEM OVERVIEW 2](#_bookmark0)
3. [SYSTEM Components 2](#_bookmark0)
   1. [Decomposition Description 3](#_bookmark1)
   2. [Dependency Description 3](#_bookmark1)
   3. Interface Description 3
      * Module Interfaces
      * User Interfaces (GUI)
4. [DETAILED DESIGN 3](#_bookmark1)
   1. [Module Detailed Design 3](#_bookmark1)
   2. [Data Detailed Design 3](#_bookmark1)
   3. RTM 3

### INTRODUCTION

## Purpose

This Software Design Document contains a comprehensive description of the structure of GitGoing, and its constituent components, including planned implementation. The expected audience is the Computer Science department of Bellevue College, including Professor Sara Farag, and other top-level Computer Science department members. Other Computer Science (CS) Department members may also find use of this document. Clients are not expected have access to this document, and thus the language is that expected of CS students/instructors.

## Scope

The basic architecture of GitGoing is a database back-end hosting a web application front-end for clients. The database is planned to be a relational database from MySQL, paired to a JavaScript (JS) web application that utilizes HTML, CSS, and React to better implement UI systems. Planned features include, but are not limited to, the ability to publish code for review, reviewing the difference in code (or possibly many types of files), commenting on said reviews, direct messaging (DM) between users, and a notification system to alert users of new comments, reviews, etc.

## Definitions, Acronyms, and Abbreviations

## CS *Computer Science*

## JS *Javascript*

## DM *Direct Message*

## VCS *Version Control System*

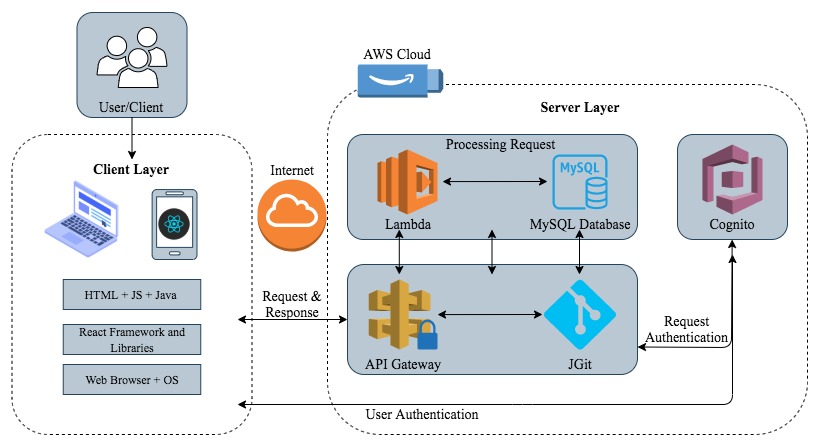
## Client A site user

## References

*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

### SYSTEM OVERVIEW



### SYSTEM Components

## Decomposition Description

## Git diff: the request/information to be sent

## Server: GET requests from API

## API Gateway: sends that request to lambda

## Lambda: takes information and stores it in the database

## Cloud Database: where requests (git diff) are stored

## Server: sends response to client

## Dependency Description

## Diagram Description automatically generated

## Interface Description

## Module Interfaces Graphical user interface, application Description automatically generated

## User Interfaces (GUI) A picture containing diagram Description automatically generated

### DETAILED DESIGN

## Module Detailed Design

## Diagram Description automatically generated Diagram Description automatically generated Diagram Description automatically generated

## Data Detailed Design

Through a MYSQL relational database, the basic design will look something like:

|  |  |  |
| --- | --- | --- |
| USERS |  |  |
|  | UID | INT(12) PK |
|  | UPW | VARCHAR(12) |
|  | Uname | VARCHAR(30) |
|  | Uemail | VARCHAR(30) |
|  | URevCnt | INT(6) |
|  | USMessages | VARCHAR(1000) |
|  | Ucomments | VARCHAR(1000) |
|  | Urmessages | VARCHAR(1000) |
|  | RevID | INT(12) |
| REVIEWS |  |  |
|  | RevID | INT(12) PK |
|  | RevName | VARCHAR(30) |
|  | RevUsers | VARCHAR(30) |
|  | RevChange | LONGBLOB |
|  | RevCount | INT(6) |
|  | RevComm | VARCHAR(1000) |
|  | RevAppro | BOOLEAN |
| ADMIN |  |  |
|  | Ucount | INT(6) |
|  | Rcount | INT(6) |
|  | Vcount | INT(6) |
|  | Mcount | INT(6) |
|  | Ccount | INT(6) |
|  | AID | INT(6) PK |

## RTM

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Req # | Requirement | Design Specification | Program Module | Test specification | Test Case Numbers | Successful test verification | Modification of requirement | Remarks |
| 1 | The user can register an account | 3.2.1 – User Registration and login system | User Client, User Authentication System |  |  |  |  |  |
| 2 | The user can verify account through email | 3.2.1 – User Registration and login system | User Client, User Authentication System |  |  |  |  |  |
| 3 | The user can login to their account | 3.2.1 – User Registration and login system | User Client, User Authentication System |  |  |  |  |  |
| 4 | The user can reset their password | 3.2.1 – User Registration and login system | User Client, User Authentication System |  |  |  |  |  |
| 5 | The user can logout of their accoun | 3.2.1 – User Registration and login system | User Client, User Authentication System |  |  |  |  |  |
| 6 | The user’s account information is secure | 3.2.1 – User Registration and login system | User Authentication System |  |  |  |  |  |
| 7 | The user is notified when assigned to a review | 3.2.2 – Notification System | User Client, Code Review System, Notification System |  |  |  |  |  |
| 8 | The user is notified when a review is approved | 3.2.2 – Notification System | User Client, Code Review System, Notification System |  |  |  |  |  |
| 9 | The user is notified when a review is rejected | 3.2.2 – Notification System | User Client, Code Review System, Notification System |  |  |  |  |  |
| 10 | The user is notified of review inquries | 3.2.2 – Notification System | User Client, Code Review System, Notification System |  |  |  |  |  |
| 11 | The user is notified of DMs | 3.2.2 – Notification System | User Client, Chat System, Notification System |  |  |  |  |  |
| 12 | The user can message other users | 3.2.3 – Chat System | User Client, Chat System |  |  |  |  |  |
| 13 | The user can add other users to a friends list | 3.2.3 – Chat System | User Client, Chat System |  |  |  |  |  |
| 14 | The user can remove a friend | 3.2.3 – Chat System | User Client, Chat System |  |  |  |  |  |
| 15 | The user can block another user | 3.2.3 – Chat System | User Client, Chat System |  |  |  |  |  |
| 16 | The user can initiate a code review | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 17 | The user can assign a reviewer | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 18 | The user is shown a diff between two file versions | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 19 | The user can leave inline comments | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 20 | The user can leave comments on the review page (not inline) | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 21 | Users assigned to reviews can approve changes | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 22 | Users assigned to reviews can reject changes | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 23 | Users assigned to a review can leave inquiries | 3.2.4 – File Change System | User Client, Code Review System |  |  |  |  |  |
| 24 | The app can handle a minimum of 10k users | 3.3 – Performance requirements | Server |  |  |  |  |  |
| 25 | Servers have enough storage space for user information and files | 3.3 – Performance requirements | Database |  |  |  |  |  |
| 26 | Chat function, notifications and comments should all perform close to real-time | 3.3 – Performance requirements | User Client, Notification System, Chat System, Code Review System |  |  |  |  |  |

### APPENDICES

*This section is optional.*

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document

### 