INDIRA GANDHI NATIONAL OPEN UNIVERSITY

OPRS : ONLINE PEER REVIEW SYSTEM

By

ADITYA PRATAP SINGH
Enrolment No: 136177847

Under The Guidance Of
Dr. Deepak Arora

www.github.com/adisingh1992/oprs

Submitted to the School of Computer and Information Sciences,

IGNOU

In Partial Fulfilment of the Requirements

for the Award of the Degree

Master Of Computer Applications (MCA)

2017

Indira Gandhi National Open University
Maidan Garhi
New Delhi – 110068

ABSTRACT

Peer review is a proven learning approach that allows students to observe and critique different solutions to a problem, as well as to receive feedback on their own work. The purpose of this project is to develop an online computer system to support this learning approach. This system will allow authors to create and submit peer review assignments easily and with a variety of configuration options. It will enable users to view the work they are reviewing electronically, submit reviews, and also display peer reviews of their own work. This report describes the system architecture, database schema, and database implementation of the first iteration of this system. It presents the client-side interface with which the users will interact, which is a Python-based client application that the users of the system (authors and reviewers) will use to upload articles, assign reviews, create reviews, provide comments etc. It concludes with our contributions, a summary of what we learned, and our plans to test the system and assess its usability and utility.

ACKNOWLEDGEMENT

"It is not possible to prepare a project report without the assistance & encouragement of other people. This one is certainly no exception." On the very outset of this report, I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have been able to make headway in this project.

I am ineffably indebted to my project-guide **Dr. Deepak Arora** for his conscientious guidance and encouragement to accomplish this project. I would also take this opportunity to thank **Mr. Ashish Shukla**, our counsellor, who helped me understand the 'underlying structures' of a project of this scale and size.

I also acknowledge with a deep sense of reverence, my gratitude towards my parents and members of my family, who have always supported me morally as well as economically. At last but not least, gratitude goes towards all of my friends who directly or indirectly helped me complete this project report. Any omission in this brief acknowledgement does not mean lack of gratitude.

[Aditya Pratap Singh]

Table of Contents

1 INTRODUCTION

1.1 BACKGROUND

There exist a number of online peer review systems. All of these systems offer the same basic functions such as interfaces for authors to submit, upload or download articles related to a journal, as well as interfaces for reviewers to review the articles, and for editors to accept or reject articles. All of these existing systems are, to some degree, not convenient to use. Moreover, none of the existing systems provide convenient interfaces to facilitate the communications between editors, users and reviewers. In addition, it is difficult to add new features to the current online journal publishing systems. There is therefore a need to develop a more flexible and convenient online journal publishing and review system to take advantage of the ever increasing technological improvements in transferring data electronically over the internet.

1.2 OBJECTIVES

The project relates in general to online peer review systems, and in particular to an online journal management and publishing solution which is designed to create a flexible, intuitive, intelligent and enterprise scale user-centered solution and is designed in a modular fashion to easily and quickly add new features and offer integration points.

This new system aims to fulfill the void which exist in current systems, The objective of this project is -

- To create a system much more user-friendly than the present products.
- To create a modular system which enables the admin to quickly add or remove new features.
- To create a system which offers the authors and reviewers much more control over their contributions than the present products.

1.3 PURPOSE, SCOPE AND APPLICABILITY

1.3.1 PURPOSE

The purpose of this project is to provide article submission, peer review, document tracking, and semi-automatic correspondence with authors and reviewers. The main reason for developing such a system resides in the need for a simple review system that fulfill the needs of a much wider audience than the current systems and can be easily adapted to any future changes required. Another important reason for developing a new system over existing one's resides in the difficulty of managing and using such systems that have been built to cater the needs of their specific domains. Looking beyond the interests of the particular stakeholders, there are three main benefits advocated for peer review:

- Improvement in the quality of published papers.
- Filtering of the output of papers to the benefit of readers.
- A '*seal of approval*' that the published work meets certain standards, in particular for lay readers.

1.3.2 SCOPE

This project comparatively offers a much more productive and efficient environment than the manual way of journal review. The scope of the project could be summarized in the following points:

- The project can be used by Universities or Private Educational Institutes to maintain and review their in-house journals or research papers.
- Being a web-based application the user can access the portal from any part of the world at any time.
- Does not require face-to-face interaction between an Author of a 'paper' and it's Reviewer. The reviewer can read and grade the 'paper' at his leisure.
- Helps the users to achieve greater productivity in lesser time.

1.3.3 APPLICABILITY

Direct-Applicability: This project directly affects the process of peer reviewing documents or papers written by authors before they could be published. It provides great efficiency to the whole excercise by greatly reducing the time required for the document to be reviewed.

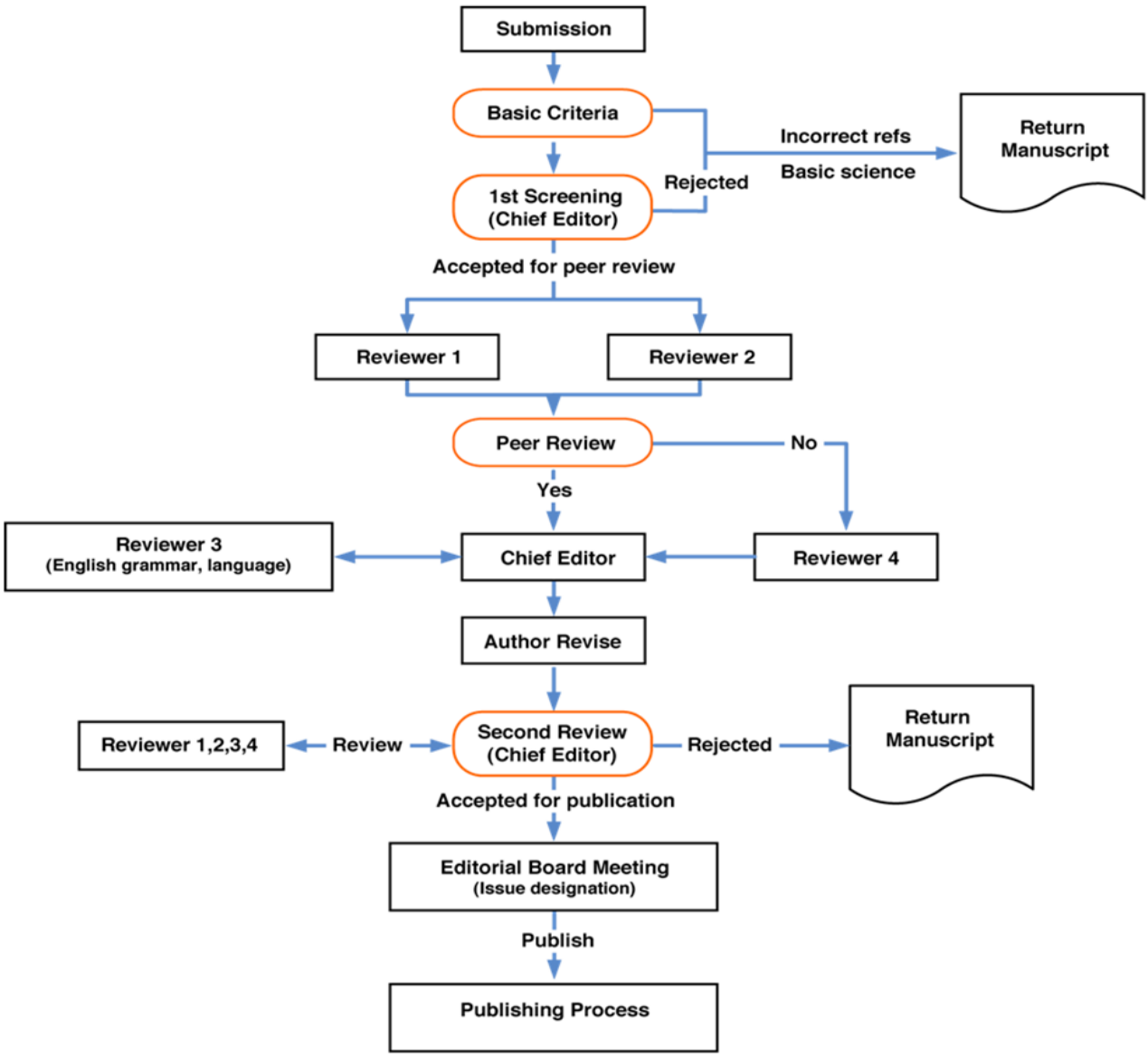


Figure 1 : Manual Process Of Peer Review

Indirect-Applicability: By providing the ease of submitting and reviewing a document online, this project encourages authors and students to proceed with their research's without worrying about the logistics of peer reviewing their 'paper'.

1.4 ACHIEVEMENTS

- This *Online Peer Review System* majorly reduces the number of steps involved in peer review thereby increasing the efficiency of the whole process.
- Encourages authors to publish more documents and papers.
- From the point of view of a developer, this system helped polish-up the details and know-how of undertaking and developing such an industry-standard specific, large scale project.

1.5 ORGANISATION OF REPORT

Chapter 1: Introduction – A brief summary of the complete system i.e. *Online Peer Review System*.

Chapter 2: Survey of Technologies – Description and details of all the technologies that are required to successfully develop and implement the system.

Chapter 3: Requirements and Analysis – This section describes all the requirements which are necessary to be fulfilled for this system to be realized like time deadlines, software/hardware requirements, database constraints etc.

Chapter 4: System Design – Comparative analysis of the architecture and design of the system. Comprehensive details of all the modules, data structures and algorithms used to implement the system are described under this section.

Chapter 5: Implementation and Testing – This chapter deals with the implementation strategies utilised like software development models, testing methodologies etc.

Chapter 6: Results and Discussions – Results based on testing done in earlier segments and user documentation to define the workings of the system is described under this chapter.

Chapter 7: Conclusions – This chapter defines the limitations of the system, points on how the system can be further improved in the future and any conclusions derived after completion of the project.

2 SURVEY OF TECHNOLOGIES

Web-based application developers face a dizzying array of platforms, languages, frameworks and technical artifacts to choose from. We survey, classify, and compare technologies supporting Web application development. The classification is based on:

- Foundational technologies
- Integration with other information sources; and
- Dynamic content generation.

2.1.1 Web Server(Light-tpd) :

Web servers implement the server-side duties of HTTP, the application-layer protocol that governs message-passing between Web clients and servers. The most common Web server implementations are the Apache HTTP server [Apache Software Foundation 2004], available for most operating systems, and the Internet Information Service (IIS) [Microsoft Corporation 2005], available only for Microsoft Windows operating systems.[2]

2.1.2 Web Browsers :

Web browsers process user interface commands, format and send request messages to Web servers, wait for and interpret server response messages, and render content within the browser's display window area.[2]

2.1.3 HTML5 AND CSS3 :

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard. It includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.[2]

HTML documents contain text interspersed with formatting elements. Cascading Style Sheets (CSS) allow formatting elements to be separated into reusable style sheets. Uneven CSS implementations in browsers and ingrained practices resulted in an extended acceptance period for style sheets. Promotion of Web standards and improved browser implementations of the standards have recently resulted in steadily increasing use of CSS to separate presentation attributes from content.

2.1.4 Client-Side Scripting :

Interpreters for lightweight scripting languages such as JavaScript and VBScript were available for most browsers by the end of 1996. Client-side scripting languages interfaces are more accessible than browser extension APIs, since they remove the need to know an entire API to add pieces of useful functionality to an HTML document. Client-side scripts are slightly less efficient than plug-ins, but the advantages of easy access to browser properties and methods outweigh the performance penalty.[2]

2.1.5 Interpreted Template-Based Scripting :

The use of templates is a common characteristic of pattern-based program construction systems. A template is a model for a set of documents, composed of fixed and variable parts, that is contextually expanded to yield conforming document instances. The variable parts of a template encapsulate programming logic such that each variable part exposes a program function that specifies how it is expanded in context. Predominately-fixed templates are easier to construct and comprehend than more variable templates since less programming is required and the defined structure is largely static. Many Web sites consist of generally fixed HTML pages with small amounts of variable content. In contrast to template processing, CGI processing is more suitable for applications with mostly variable content, which are not as common on the Web. The template model better supports the common Web application pattern by embedding logic within fixed presentation markup.[2]

2.1.6 Python :

Python is a widely used high-level programming language for general-purpose programming, created by **Guido van Rossum** and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Picture : Python Logo™

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.[3]

3 REQUIREMENTS AND ANALYSIS

3.1 PROBLEM DEFINITION

Peer review, known as refereeing in some academic fields, is a process of subjecting an author's scholarly work, research or ideas to the scrutiny of others who are experts in the same field. In this report we will consider only the peer review of articles submitted to academic journals.

In journals peer review, the author's article is usually subjected to some initial checks to assess its suitability for review (for instance, incomplete articles or work that was patently pseudoscience would be declined without review), after which a small number of reviewers are selected. The task expected of the reviewers varies somewhat from journal to journal, but in essence, is usually to assist the journal's editor (who makes the final decision) on deciding whether or not to accept the article for publication. The reviewer will comment on the quality of the work done as well as on its originality and its importance.

3.2 REQUIREMENTS SPECIFICATION

Various requirements of the system are categorized under following sub-headings:

PERFORMANCE REQUIREMENTS -

- Loading speed of all web-pages should be under 2 to 3 seconds.
- Any operation based on Database should not take more than 5 seconds.

SECURITY REQUIREMENTS – Security standards to be followed while developing the project are as follows:

- Used Id's and Passwords of user's must be hashed using one-way hashing algorithms before storing them into the database.
- Appropriate methods like *CAPTCHA's* should be implemented to prevent machine brute-force attacks.

- Any input by user must be validated and sanitized for preventing SQL Injection attacks.
- A well-defined password policy must be adopted, including password change frequency, invalid attempts, etc., to provide optimal level of security to user and application data.

AVAILABILITY AND RELIABILITY REQUIREMENTS - The portal should be designed to be available no less than 99.99% of time. All upgrades to the system including software updates, patches and fixes must be done without taking the portal offline.

MAINTAINABILITY -

- Every code module should have a well documented manual including logic.
- All code components should be thoroughly tested and the test coverage should be more than 80%.

DESIGN CONSTRAINTS -

The *Online Peer Review System* project must adhere to the following standards or protocols:

- Web pages are to be designed using HTML-5 and formatted using CSS-3 transitional standards.
- W3C Web Accessibility standards, Web Content Accessibility Guidelines, should be followed including keyboard navigation, alternate titles for images, etc.

3.3 PLANNING AND SCHEDULING

The product is scheduled to be completed within 1.8 months or around 51 days. A complete chart of the process or activities to be carried-out is given below which is divided into two parts,

- *Project Evaluation & Review Technique* chart.
- *GANTT* chart (Named after Henry L. Gantt).

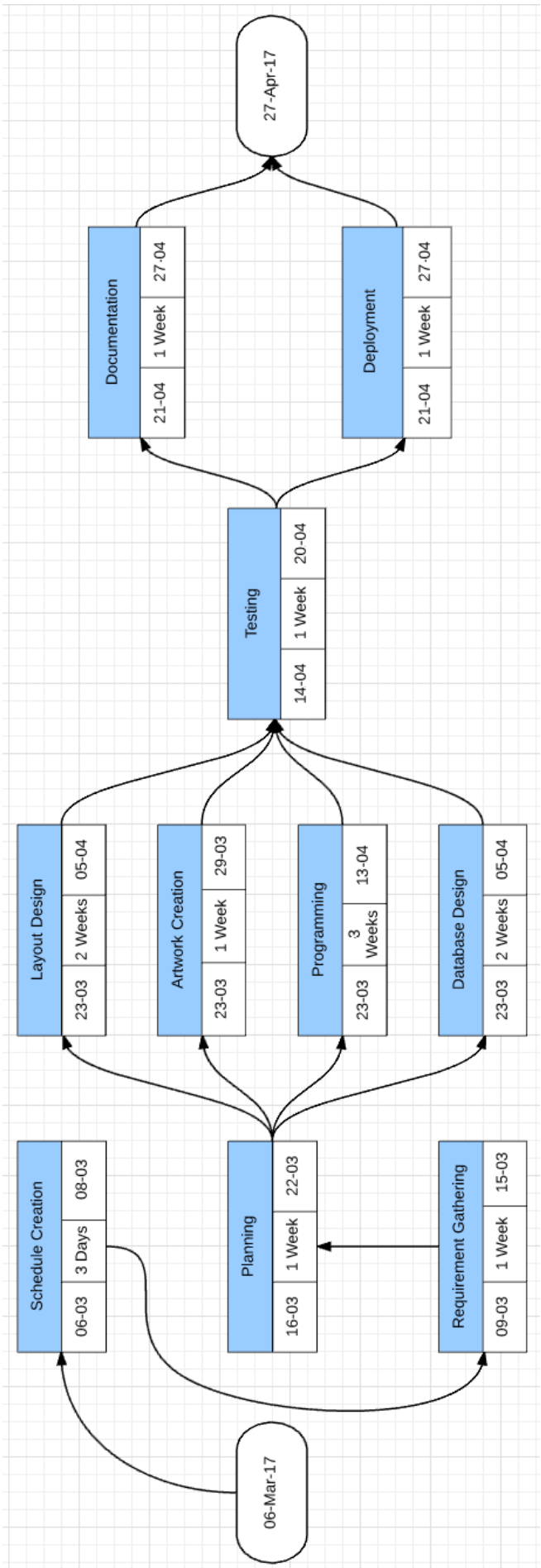


Figure 2 : Project Evaluation And Review Technique Chart

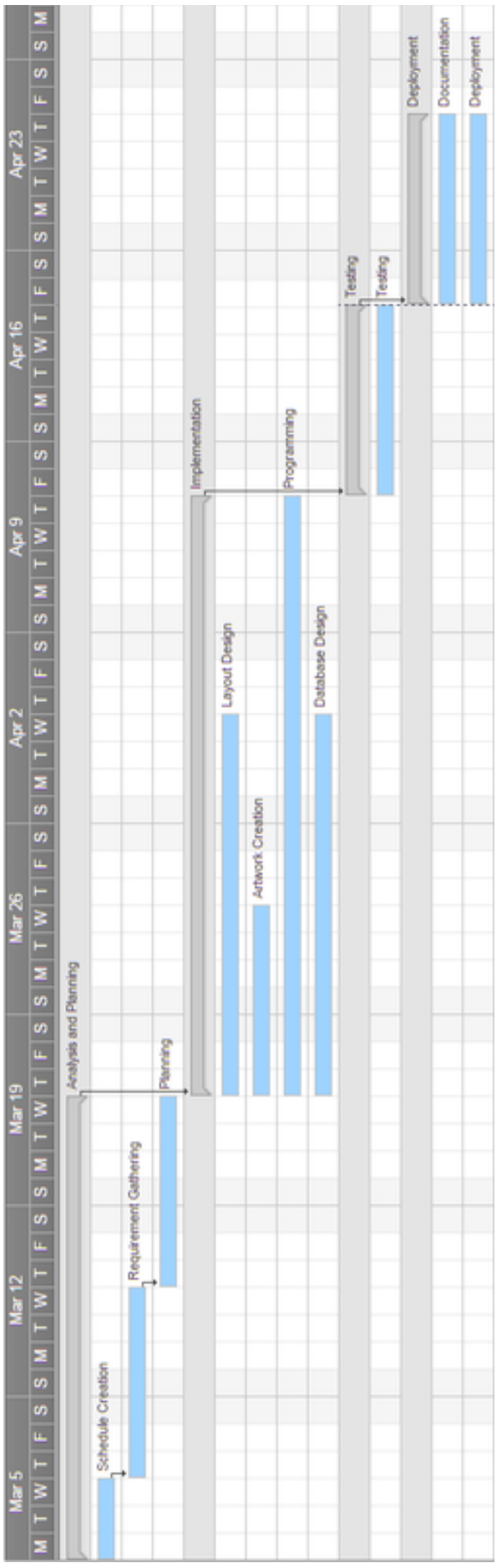


Figure 3 : Gantt Chart

3.4 SOFTWARE AND HARDWARE REQUIREMENTS

3.4.1 Minimum Hardware Requirements (SERVER) -

- Pentium Class or equivalent processor.
- A Gigabyte of RAM.
- 0.5 Gigabyte of free disk space for web-server and software source.
- Minimum 0.5 Gigabyte of Hard-Disk space for database.

3.4.2 Recommended Hardware Requirements (SERVER) -

- Intel Multi-Core processor or equivalent.
- 2 Gigabyte of RAM.
- 0.5 Gigabyte of free disk space for web-server and software source.
- Dynamically allocated storage space for Database according to database size.

3.4.3 Software Requirements (SERVER) -

- Linux or UNIX based operating environment.
- Python 2.7 Interpreter.
- Flask framework.
- Light-tpd (Apache based Web-Server).
- MySQL (Database Server).

3.4.4 Hardware/Software Requirements (CLIENT) -

- Any system capable of running a web-browser.
- Web-browser must support javascript processing for smooth working of the webapp.
- A working internet connection.

3.5 PRELIMINARY PRODUCT DESCRIPTION

The *O.P.R.S.* web-app classifies it's users into following three categories:

- **REGISTERED USER –** A registered user has access to all the articles published on the web-app including the comments made by the reviewers. This type of user can also submit articles and comments based on his expertise or area of subject he has chosen during registration.
- **UNREGISTERED USER –** A user who lands on the wep-app for the first time is required to register himself with the portal to continue to view the articles published.

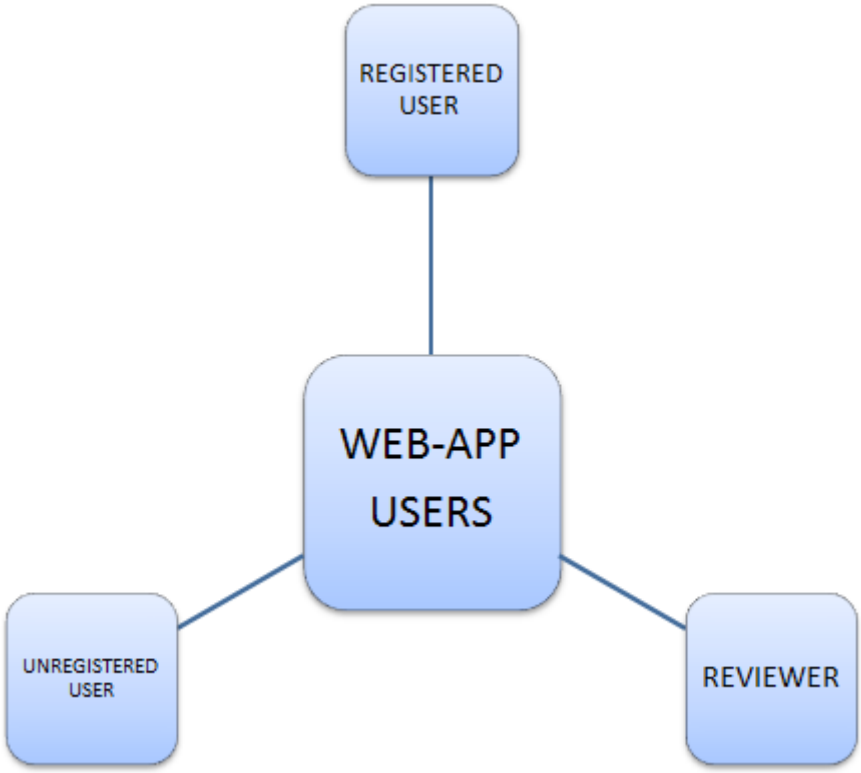


Illustration 1 : Assumed Users

- **REVIEWER –** A reviewer has access to all the articles published or submitted for review by a registered user. While the reviewer can access all the articles, he is only allowed to review and comment on articles of the subject he is an expert in. He would not be allowed to review an article outside of his area of expertise.

3.6 CONCEPTUAL MODELS

3.6.1 DATA FLOW DIAGRAMS :

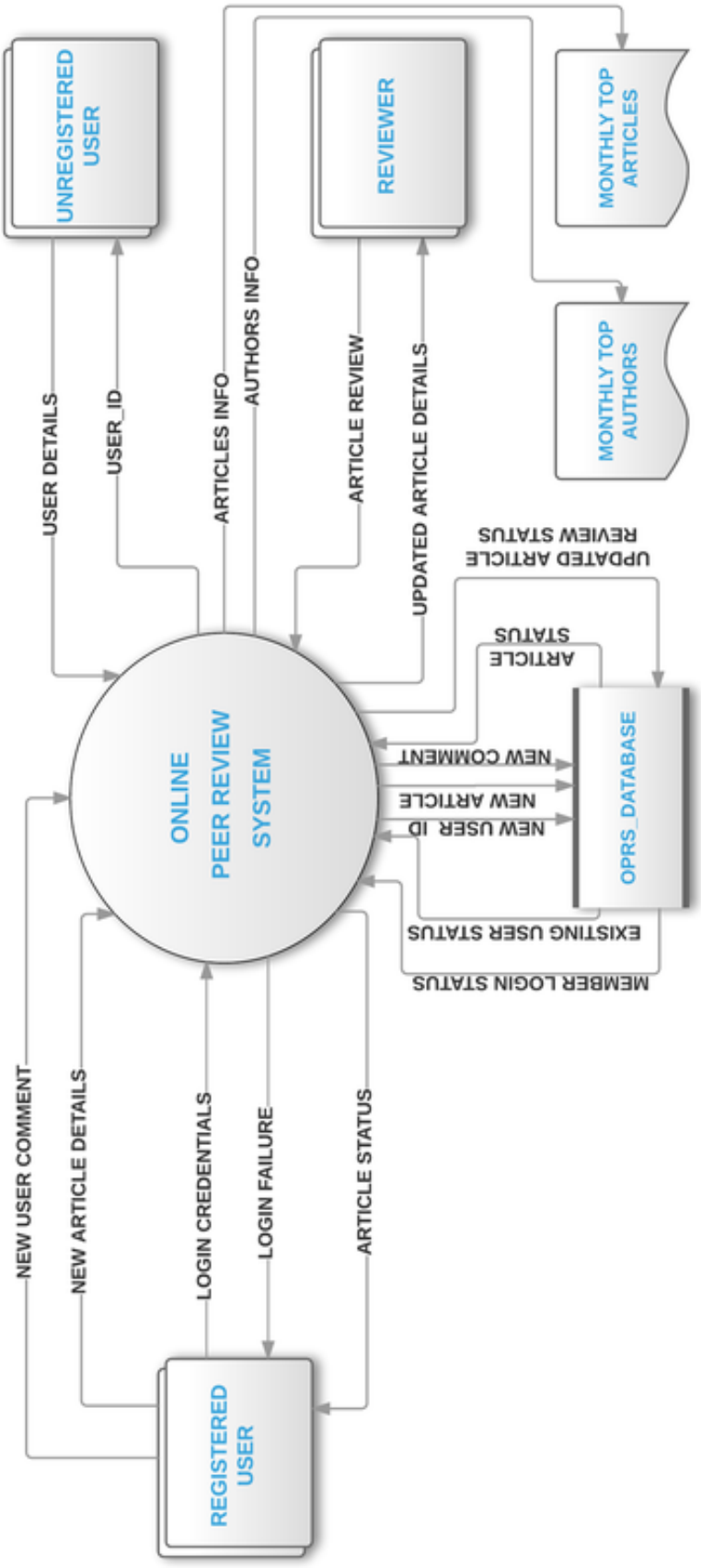


Figure 4.1 : Context Level DFD

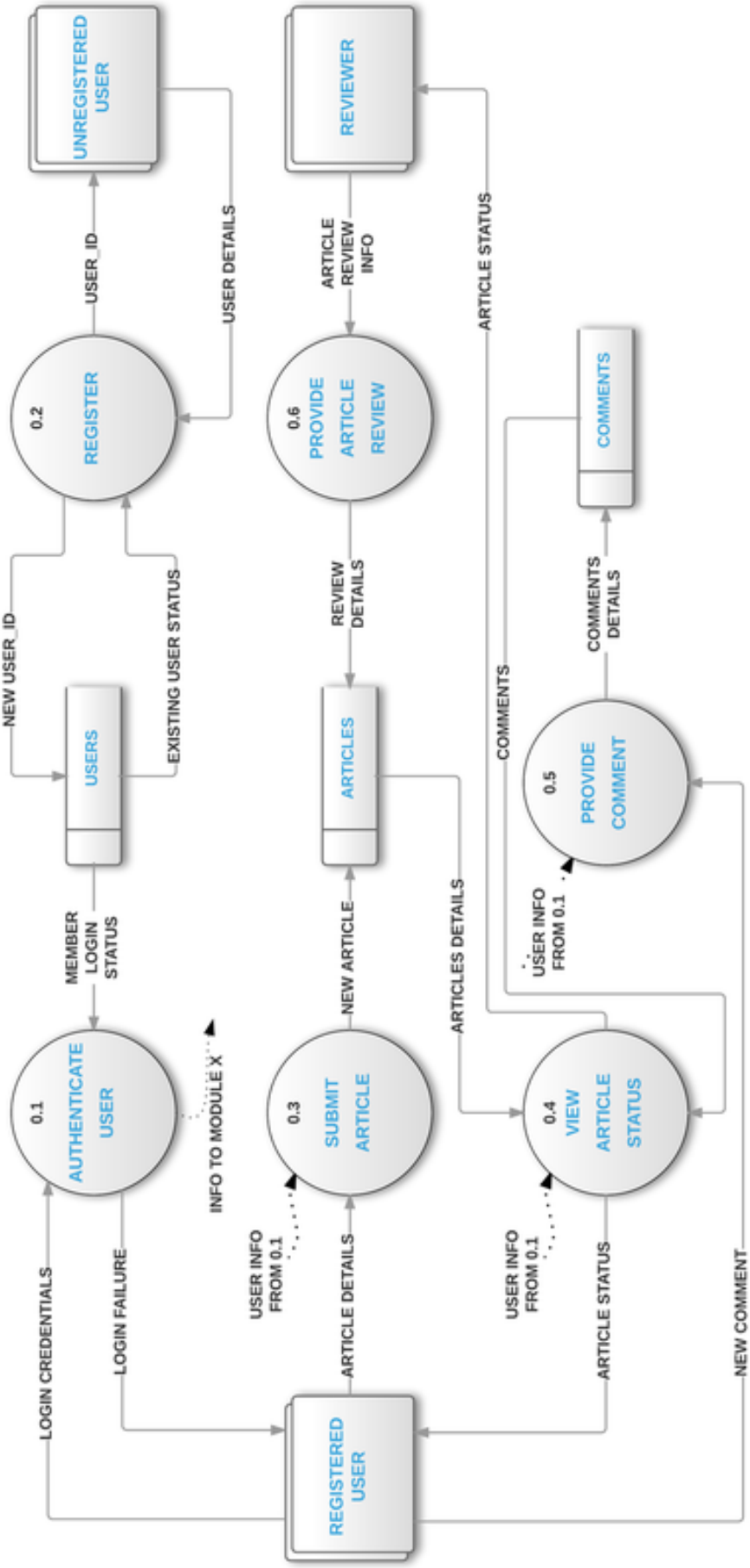


Figure 4.2 : First Level DFD

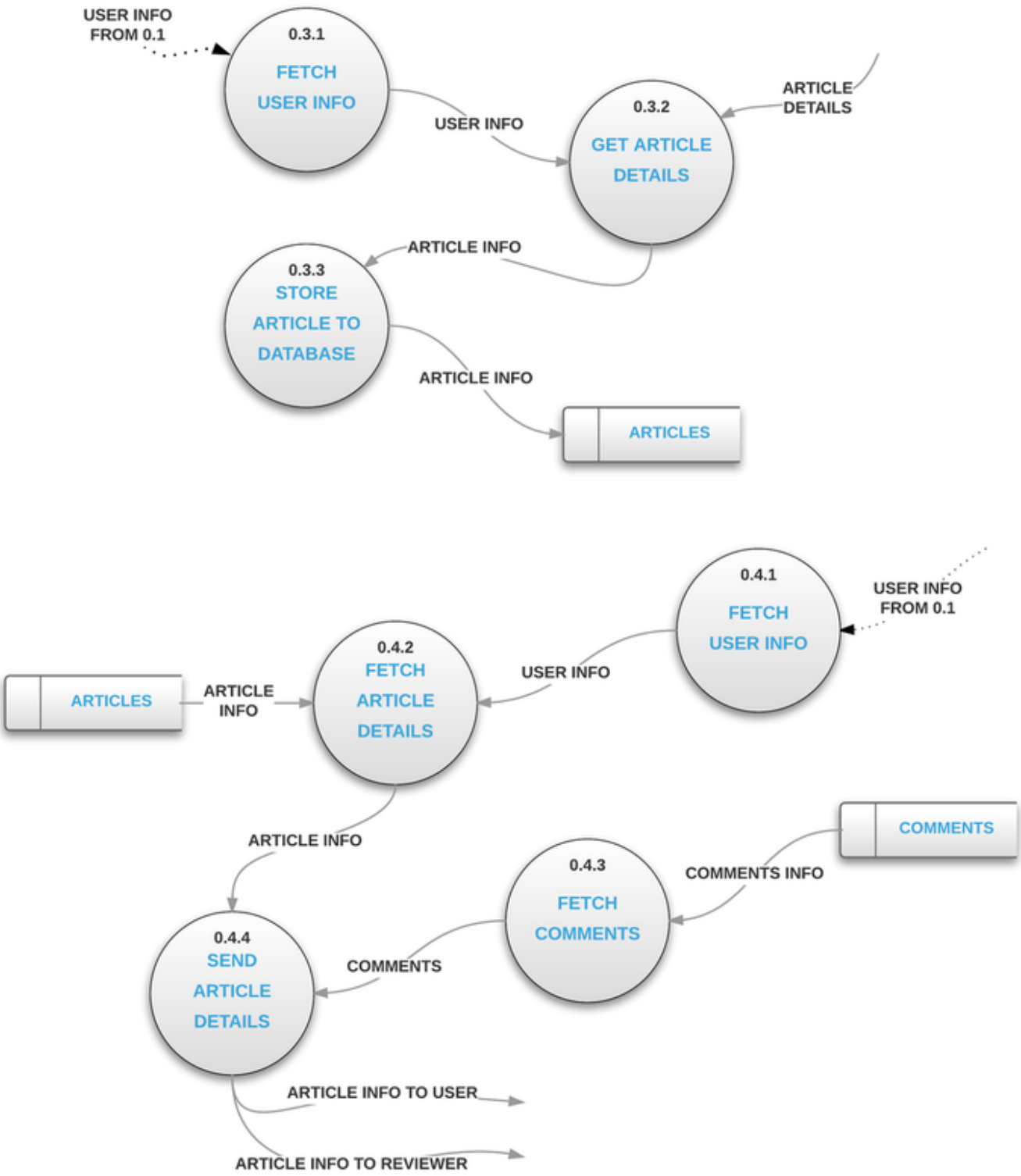


Figure 4.3 : Second Level DFD

3.6.2 ENTITY RELATIONSHIP DIAGRAM :

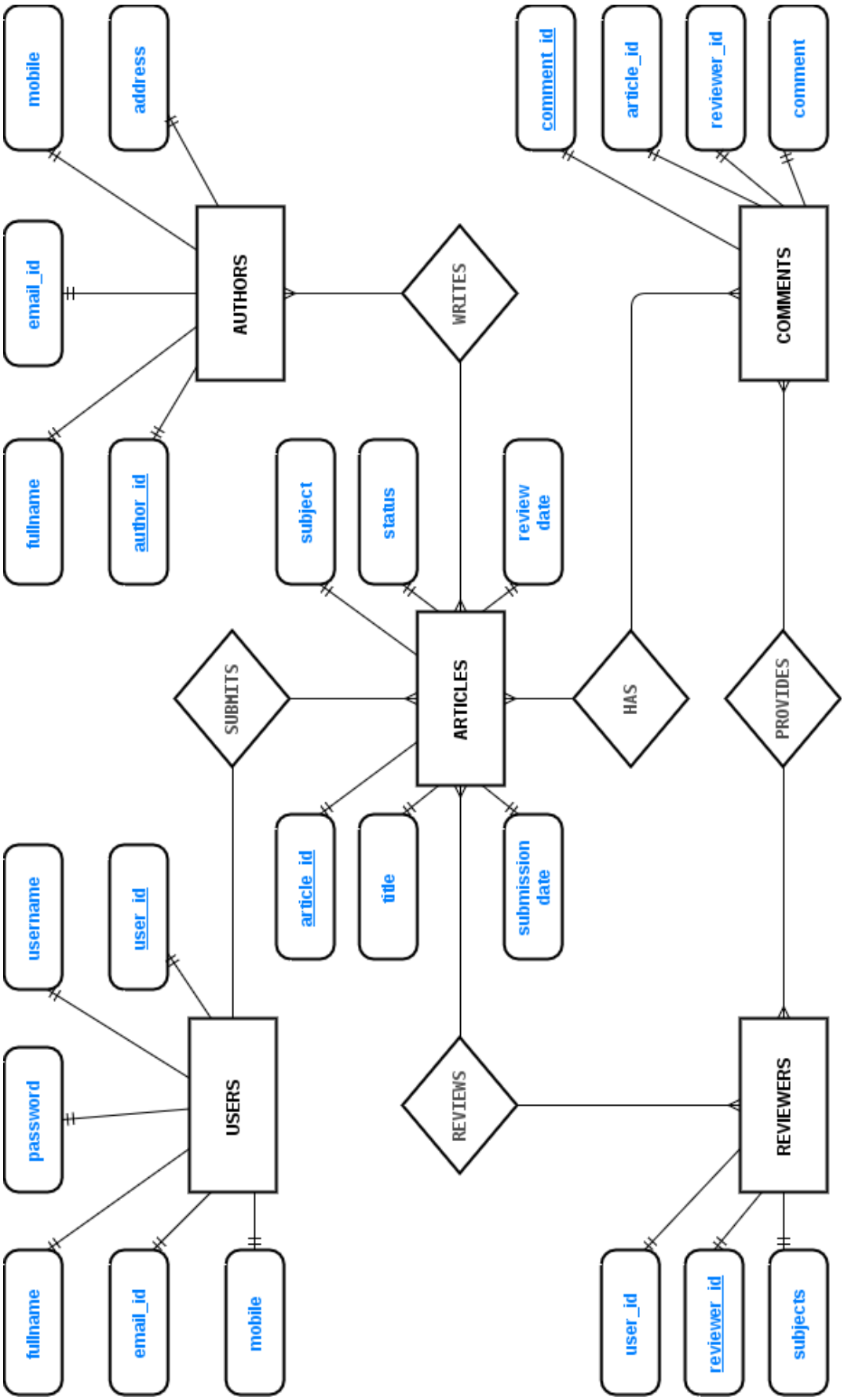


Figure 5 : Entity Relationship Diagram

4 SYSTEM DESIGN

4.1 BASIC MODULES

The web-app is divided into various modules, they are:

REGISTRATION MODULE - This module allows a first-time visiting user to register his/her credentials into the portal's database. Upon successful registration they will be redirected to the *authentication* module where they can provide their credentials to enter the users module to view the articles.

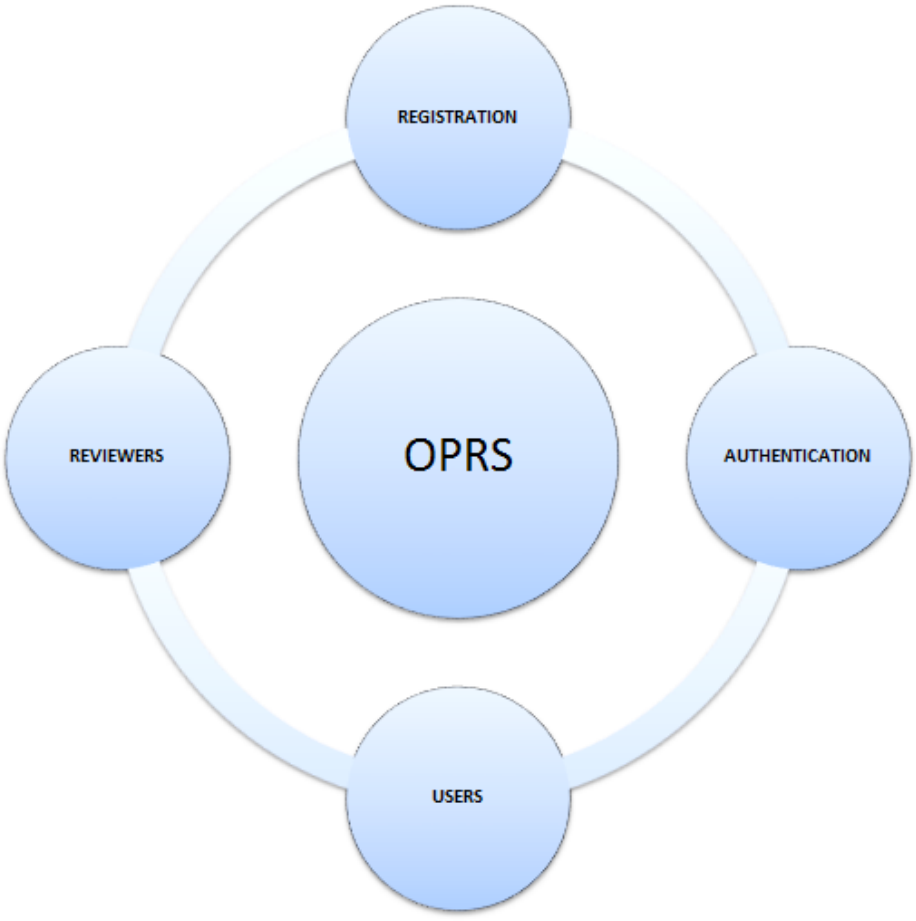


Illustration 2 : Project Modules

AUTHENTICATION MODULE – This module is divided into two subparts, upon recieving and verifying the credentials, the user based on his type is redirected to the users module or the reviewers module.

USERS MODULE – A user with no article reviewing rights are redirected to this module. The articles which are *ACCEPTED* by the reviewers are displayed here. Under this module, reviewer comments are also displayed under the articles.

Picture : Modules

REVIEWERS MODULE – This module is an extention to the users module. Users who possess article reviewing rights are redirected here. They can *ACCEPT*, *REJECT* or send back an article for *MODIFICATION*. In addition to viewing comments made by other reviewers they can provide their own, to articles they have expertise in.

COMMENTS MODULE – Comments module allows the registered users including the reviewers to provide comments on an article of their field of expertise. A user must be authenticated to comment on an article.

4.2 DATA DESIGN

4.2.1 DATABASE DESIGN :

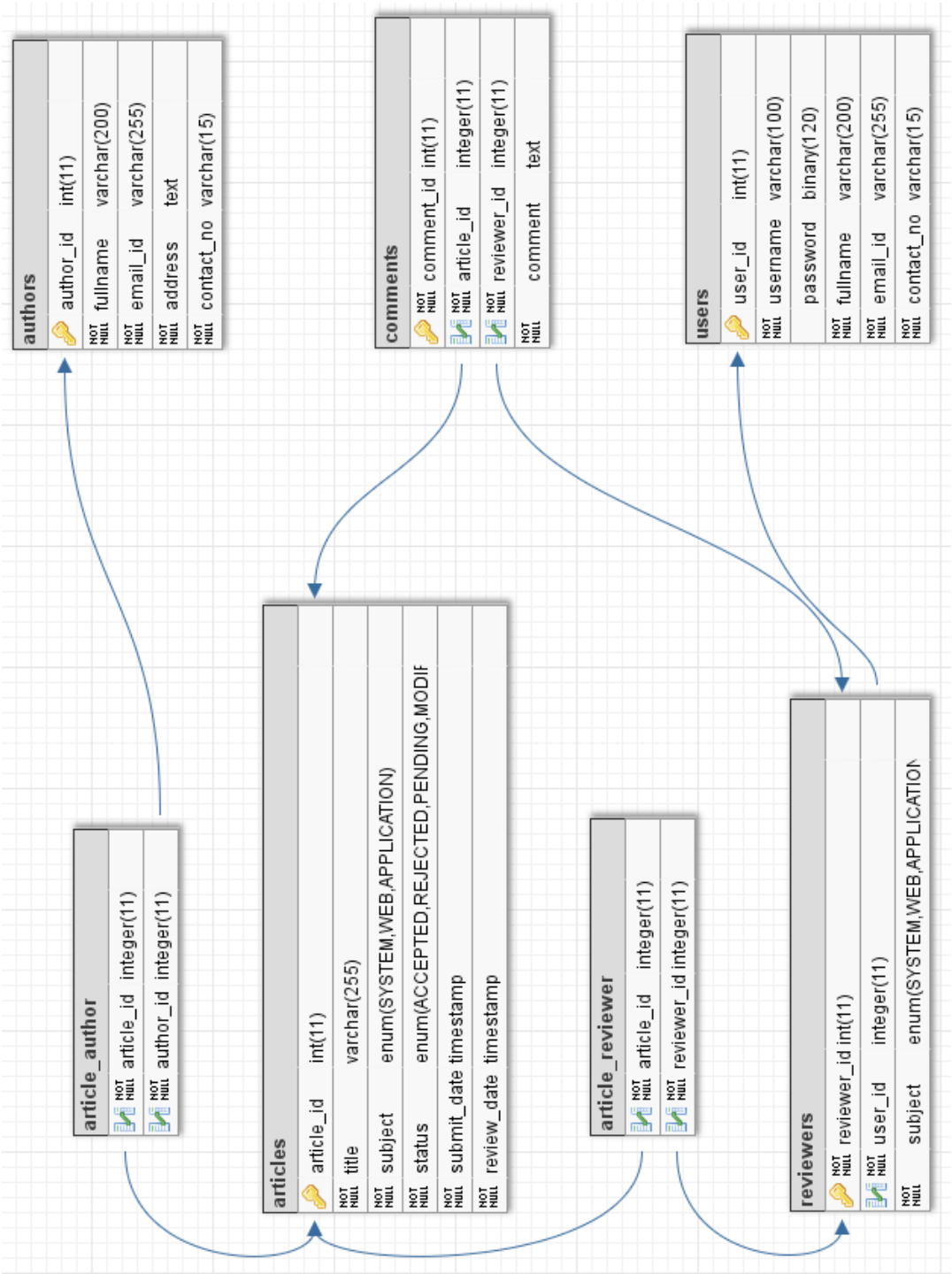


Figure 6 : Database Design Diagram

4.2.2 DATA DICTIONARY :

USERS

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(100)	NO		NULL	
password	binary(64)	NO		NULL	
fullname	varchar(200)	NO		NULL	
email_id	varchar(255)	NO		NULL	
contact_no	varchar(15)	NO		0000000000	

AUTHORS

Field	Type	Null	Key	Default	Extra
author_id	int(11)	NO	PRI	NULL	auto_increment
fullname	varchar(200)	NO		NULL	
email_id	varchar(255)	NO		NULL	
address	text	NO		NULL	
contact_no	varchar(15)	NO		0000000000	

COMMENTS

Field	Type	Null	Key	Default	Extra
comment_id	int(11)	NO	PRI	NULL	auto_increment
article_id	int(11)	NO	MUL	NULL	
reviewer_id	int(11)	NO	MUL	NULL	
comment	text	NO		NULL	

REVIEWERS

Field	Type	Null	Key	Default	Extra
reviewer_id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO	MUL	NULL	
subject	enum('SYSTEM','WEB','APPLICATION')	NO		NULL	

ARTICLES

Field	Type	Null	Key	Default	Extra
article_id	int(11)	NO	PRI	NULL	auto_increment
title	varchar(255)	NO		NULL	
subject	enum('SYSTEM','WEB','APPLICATION')	NO		NULL	
status	enum('ACCEPTED','REJECTED','PENDING','MODIFY')	NO		PENDING	
submit_date	timestamp	NO		CURRENT_TIMESTAMP	
review_date	timestamp	NO		0000-00-00 00:00:00	

ARTICLE_AUTHOR (Look-up Table)

Field	Type	Null	Key	Default	Extra
article_id	int(11)	NO	MUL	NULL	
author_id	int(11)	NO	MUL	NULL	

ARTICLE_REVIEWER (Look-up Table)

Field	Type	Null	Key	Default	Extra
article_id	int(11)	NO	MUL	NULL	
reviewer_id	int(11)	NO	MUL	NULL	

4.3 PROCEDURAL DESIGN

The *Online Peer Review System* works on multiple processses which goes through different states complete a given task. A general state diagram showing the review process which is followed by the '*Review Module*' is as follows:

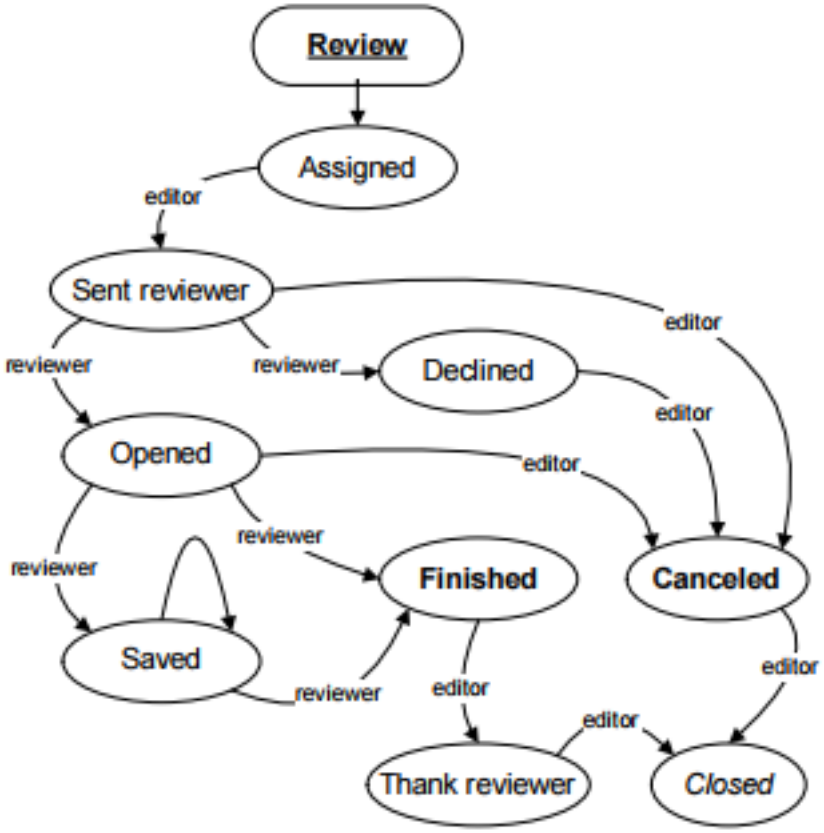


Figure 7 : Review State Diagram

According to the above diagram, once an article is submitted it goes through multiple states, they are:

- A reviewer is assigned to review the article.
- Then the reviewer, after going through the article, *approves* or *rejects* the article.
- Then the reviewer provides his feedback through the comments section.
- If the article is rejected it is sent back to the author to make necessary changes.

A top-level overview of the complete review process is depicted in the following process flow chart. It shows the complete journey of an article from it's submission to it's approval or rejection.

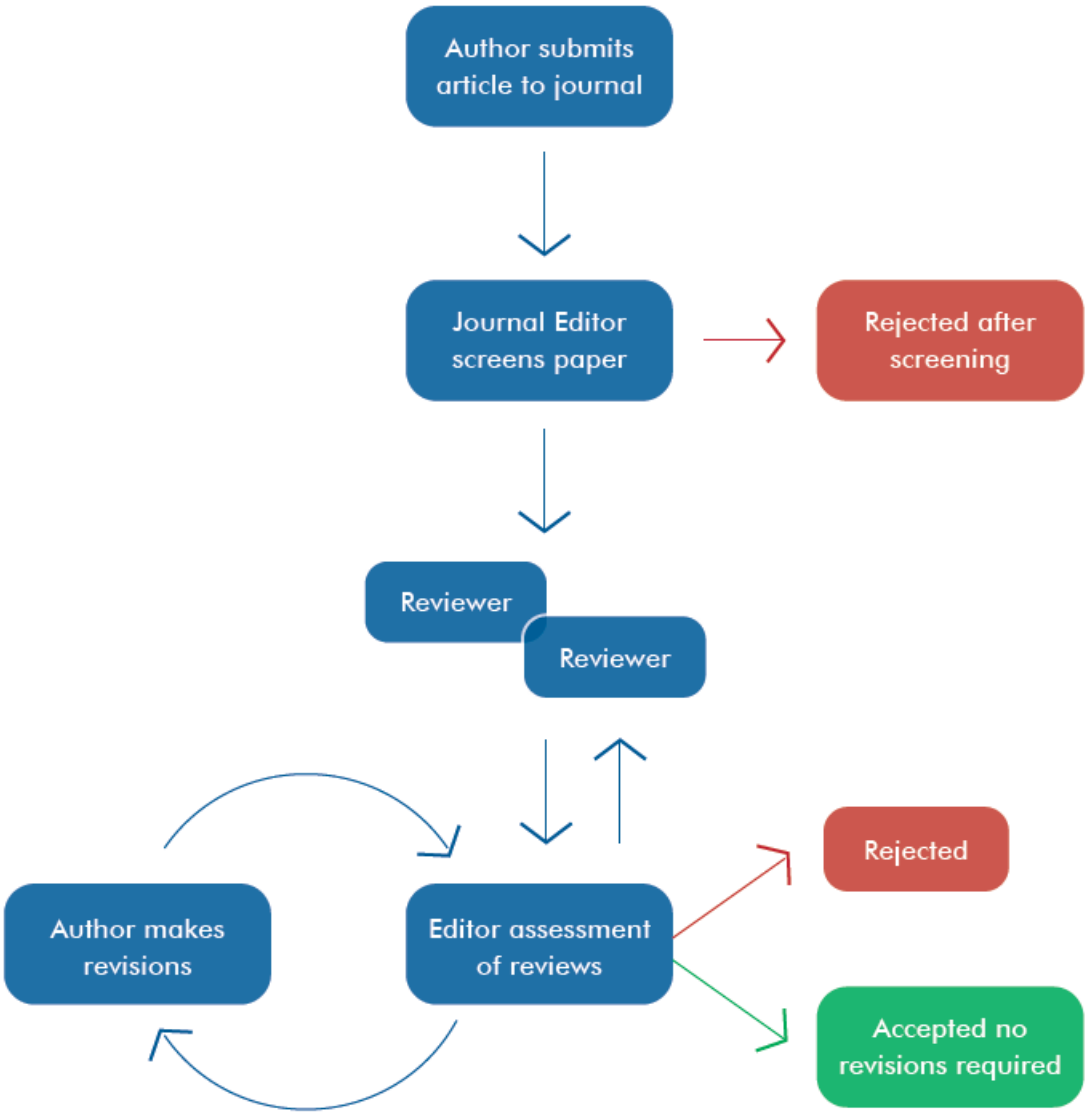


Figure 8 : Peer Review Flow Chart

4.4 USER INTERFACE DESIGN

There are a number of activities performed for designing user interface. The process of GUI design and implementation is alike SDLC. Any model can be used for GUI implementation among Waterfall, Iterative or Spiral Model. A model used for GUI design and development should fulfill the GUI specific steps described in the below illustration:

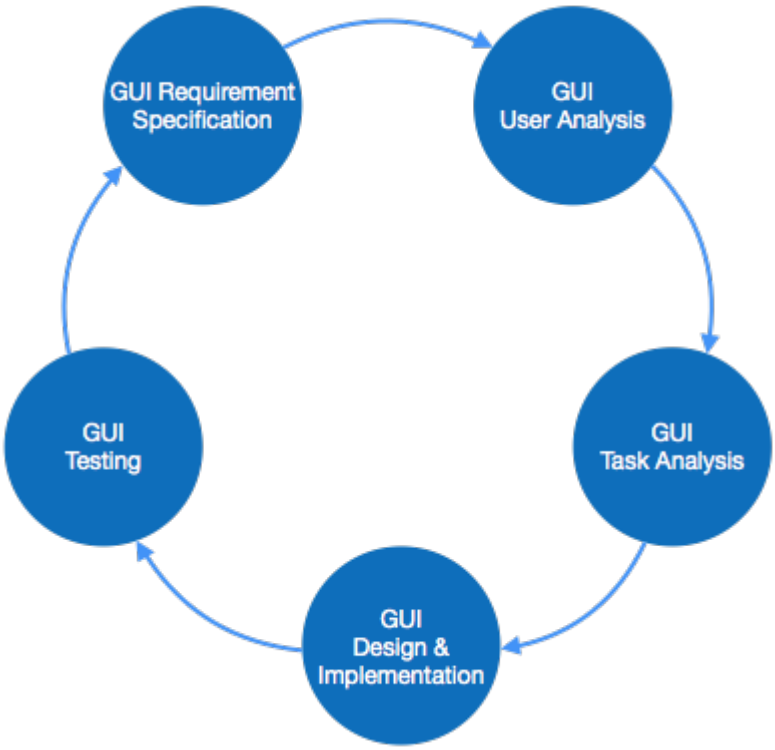


Illustration 3 : User Interface Design Process

The main user interface components of this project, '*Online Peer Review System*' are as follows:

- Home Page
- User Authentication Page
- User Registration Page
- Article Submission Page
- Comments Page
- Reviewers Page

4.5 SECURITY ISSUES

Any online system faces major security threats at any given point of time. The '*Online Peer Review System*' while being open to all to view the submitted articles, must also provide comprehensive security and privacy to it's registered users.

Cross-Site Scripting (XSS) – The project has multiple input forms which could be subjected to XSS or cross-site scripting by malicious users. In order to prevent it any kind of user-input must be properly sanitized before being used by the system. The project follows multiple steps to prevent XSS, they are:

- **Client-Side Validation:** Before even the form is submitted, the data is verified for it's integrity using various techniques with the help of javascript.
- **Server-Side Validation:** Once the form is submitted and the data is sent to the server for processing, it again gets checked to verify and validate if it is somehow manipulated in the transfer or not. After the sanitization process is complete then only the data can be further processed.

SQL-Injection – This kind of attack targets the database of a system. It forces the backend database to give out information to a user who may not have proper privileges to the system. To prevent sql-injection attacks, the system follows a number of steps, they are:

- **User-Input Sanitization –** Any kind of input from client-side is checked and cleaned of any non-allowed characters which can be used in sql-injection and other types of attacks. No sql command in form of user input is accepted by the system.
- **Prepared Statements –** On the server-side, prepared statements are used to run database queries so as to allow only those parameters to be passed to the database which are necessary and no other subsequent queries are allowed as part of the parameters.

Session Hijacking – Proper '*sessions*' protocols has been implemented to prevent one user from acquiring the identity of another user and there by creating a security risk.

4.6 TEST CASES DESIGN

TC01 – Login Page : Authenticate successfully on *Peer Review* website.

Description – A registered user should be able to successfully login at *oprs.com*.

Precondition – The user must already be registered with an email address, username and password.

Assumption – A supported browser is being used.

Test Steps –

1. Navigate to *oprs.com/login*.
2. In the '*username*' field, enter the username.
3. In the '*password*' field enter the password.
4. Click the '*Sign-in*' button.

Expected Result – A page displaying all the articles submitted to the system should open.

==========

TC02 – Registration Page : Register successfully on *Peer Review* website.

Description – A user should be able to successfully register at *oprs.com*.

Precondition – The user must poses a valid email address.

Assumption – A supported browser is being used.

Test Steps –

1. Navigate to *oprs.com/register*.
2. Enter the personal information like Fullname, Contact Number, etc.
3. In the username and password field provide a unique username and password.
4. Password should be more than 8 characters.
5. Select the '*reviewer*' option to chose a subject of expertise.

6. Accept 'Terms of Service' by checking the respective box.
7. Click on '*Register*'.

Expected Result – The user should be redirected to the login page upon successful registration.

==========

TC03 – Upload Article : Upload an article successfully on *Peer Review* website.

Description – A user should be able to successfully upload an article at *oprs.com*.

Precondition – The user must already be registered with a username and password.

Assumption – A supported browser is being used.

Test Steps –

1. Navigate to *oprs.com/uploads*.
2. Enter the information like Author name, E-mail address, Contact Number, Title of the article, Article Subject etc.
3. Click on the '*Choose File*' button to select the article file.
4. Click on '*Submit*' button.

Expected Result – The article should be uploaded successfully and a message be displayed to the user.

==========

TC04 – Comment Form : Provide a comment successfully on *Peer Review* website.

Description – A user should be able to comment successfully on an article at *oprs.com*.

Precondition – The user must already be registered with a username and password.

Assumption – A supported browser is being used.

Test Steps –

1. Navigate to *oprs.com/view*.
2. Move to the desired article on which the comment is to be provided.
3. Click on the '*Text area field*' and enter the comment.
4. Click on '*Submit*' button.

Expected Result – The comment should be posted successfully and a message be displayed to the user.

==========

TC05 – Review Form : Provide a review successfully on *Peer Review* website.

Description – A reviewer should be able to successfully review an article at *oprs.com*.

Precondition – The user must already be registered with a username and password.

Assumption – A supported browser is being used.

Test Steps –

1. Navigate to *oprs.com/view*.
2. Move to the desired article on which the comment is to be provided.
3. Choose a feedback option from the dropdown menu.
4. Click on '*Submit*' button.

Expected Result – The review should be posted successfully and a message be displayed to the reviewer.

==========

5 IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION APPROACHES

For the development of this project, *ITERATIVE INCREMENTAL MODEL* has been used. It allows the whole project to be divided into various builds. Under this model, each module passes through the *Requirements*, *Analysis*, *Design* and *Implementation* phases. A working version of the software is produced after each cycle thereby providing a working software early-on during the *Software Development Life Cycle.* After that, each cycle adds function to the previous release. The process continues till the complete system is achieved.[11]

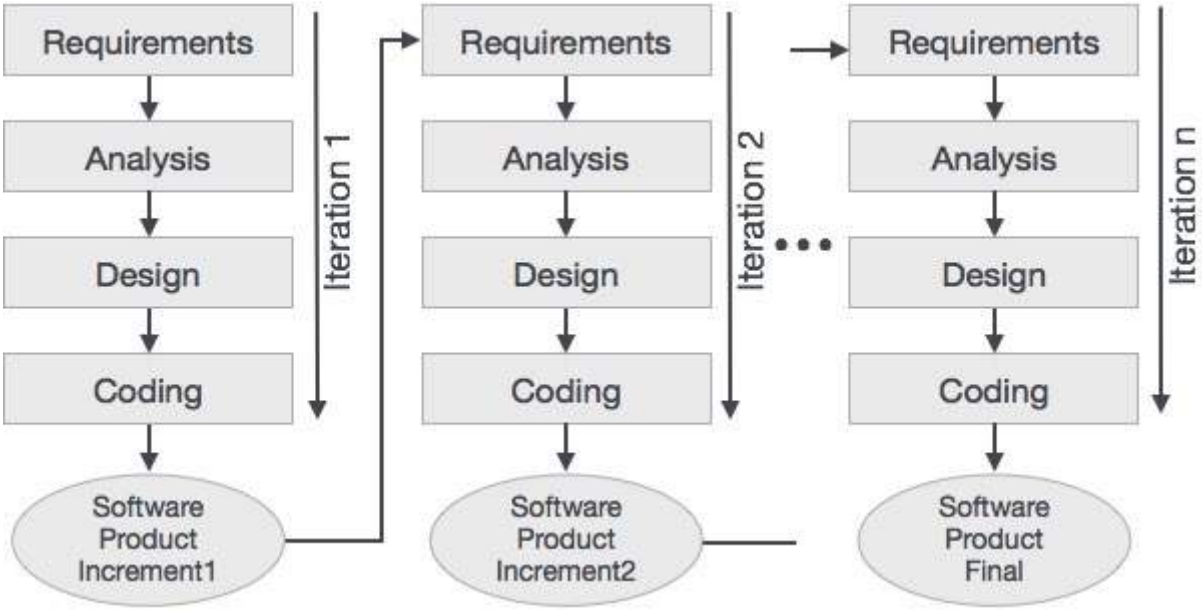


Figure 9 : Iterative Incremental Model

WHY THIS MODEL ? : This SDLC model has been used for this project as -

- It is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during each iteration.
- Generates working software quickly and early during the software life cycle.
- Lowers initial delivery cost.

5.2 CODING DETAILS AND CODE EFFICIENCY

5.2.1 PROGRAM CODE

__init__.py

```
##########################################
##Imports
##########################################
from flask import Flask
from flask import flash
from flask import redirect
from flask import render_template
from flask import request
from flask import session
from flask import url_for
from flask_mail import Mail
from flask_mail import Message
from forms import UploadFile
from forms import PassReset
from forms import LoginForm
from forms import ReviewForm
from forms import CommentForm
from forms import RegistrationForm
from functions import commentArticle
from functions import viewTopComments
from functions import reviewArticle
from functions import viewArticle
from functions import listArticles
from functions import uploadArticle
from functions import processReset
from functions import processLogin
from functions import resetVerifyCheck
```

```
from functions import processRegistration
##########################################

app = Flask(__name__)
app.config['MAX_CONTENT_LENGTH'] = 5 * 1024 * 1024

##########################################
##Mail Configuration
##########################################
mail = Mail()
app.config["MAIL_SERVER"] = "smtp.gmail.com"
app.config["MAIL_PORT"] = 465
app.config["MAIL_USE_SSL"] = True
app.config["MAIL_USERNAME"] = 'abcd@gmail.com'
app.config["MAIL_PASSWORD"] = 'abcdefgh'
mail.init_app(app)
##########################################

##########################################
##Views
##########################################
@app.route('/')
def index():
    comments = viewTopComments()
    return render_template('index.html', comments=comments)

@app.route('/home/')
def home():
    if 'logged_in' not in session:
        flash("You Have To Login First To Access This Page..!!")
        return redirect(url_for('login'))
```

```
    articles, excerpts = listArticles()
    length = len(articles)
    return render_template('home.html', articles=articles, excerpts=excerpts,
length=length)

@app.route('/register/', methods=['GET', 'POST'])
def register():
    if 'logged_in' in session:
        flash("You're Already Logged-In..!!")
        return redirect(url_for('home'))
    form = RegistrationForm(request.form)
    reg_result = ''
    if request.method == 'POST' and form.validate():
        reg_result = processRegistration(form)
        if reg_result == 0:
            flash('Thanks For Registering..!!')
            return redirect(url_for('login'))
    return render_template('register.html', form=form, error=reg_result)

@app.route('/login/', methods=['GET', 'POST'])
def login():
    if 'logged_in' in session:
        flash("You're Already Logged-In..!!")
        return redirect(url_for('home'))
    form = LoginForm(request.form)
    if request.method == 'POST' and form.validate():
        login_result = processLogin(form)
        if login_result == True:
            return redirect(url_for('home'))
        else:
            flash(login_result)
```

```
    return render_template('login.html', form=form)


@app.route('/logout/')
def logout():
    if 'logged_in' in session:
        session.clear()
        flash("You've Been Successfully Logged-Out..!!")
    else:
        flash("You're Not Logged In..!!")
    return redirect(url_for('index'))


@app.route('/reset/', methods=['GET', 'POST'])
def passReset():
    if 'logged_in' in session:
        flash("You're Already Logged-In..!!")
        return redirect(url_for('home'))
    form = PassReset(request.form)
    if request.method == 'POST' and form.validate():
        reset_result, email, token = processReset(form)
        if email is not None and reset_result is None:
            try:
                msg = Message("Password Reset (OPRS)",
sender='adi.singh1992@gmail.com', recipients=[email])
                token = "Go To The Following URL To Succesfully Change Your
Password..!!:- localhost:5000/reset/%s" %token
                msg.body = token
                mail.send(msg)
                flash("Follow The Link Sent To Your E-mail Account To Complete The
Process Successfully..!!")
            except:
                flash("Oops, Something Went Wrong, Please Try Again..!!")
```

```
        else:
            flash(reset_result)
    return render_template('reset.html', form=form)

@app.route('/reset/<key>')
def resetVerify(key):
    verify_result = resetVerifyCheck(key)
    flash(verify_result)
    return redirect(url_for('login'))

@app.route('/uploads/', methods=['GET', 'POST'])
def uploads():
    if 'logged_in' not in session:
        flash("You Have To Login First To Access This Page..!!")
        return redirect(url_for('login'))
    form = UploadFile(request.form)
    if request.method == 'POST' and form.validate():
        file_obj = request.files
        upload_status = uploadArticle(form, file_obj)
        flash(upload_status)
    return render_template('uploads.html', form=form)

@app.route('/view/<filename>', methods=['GET', 'POST'])
def displayArticle(filename):
    if 'logged_in' not in session:
        flash("You Have To Login First To Access This Page..!!")
        return redirect(url_for('login'))
    details, article, status, comment = viewArticle(filename)
    if status == True:
        return render_template('view.html', form=ReviewForm(),
form1=CommentForm(), details=details, article=article, comment=comment)
```

```
    flash(status)
    return redirect(url_for('home'))

@app.route('/review/<filename>', methods=['POST'])
def review(filename):
    if 'logged_in' and 'reviewer' not in session:
        flash("You Are Not Authorized To Access This Page..!!")
        return redirect(url_for('login'))
    form = ReviewForm(request.form)
    if form.validate():
        review_result = reviewArticle(form, filename)
        flash(review_result)
    return redirect(request.referrer)

@app.route('/comment/<filename>', methods=['POST'])
def comment(filename):
    if 'logged_in' and 'reviewer' not in session:
        flash("You Are Not Authorized To Access This Page..!!")
        return redirect(url_for('login'))
    form = CommentForm(request.form)
    if form.validate():
        comment_result = commentArticle(form, filename)
        flash(comment_result)
    return redirect(request.referrer)

@app.route('/comments/')
def comments():
    comments = viewTopComments()
    return render_template('comments.html', comments=comments)
##########################################
```

```
########################################
##Application Initialization
########################################
if __name__ == "__main__":
    app.secret_key =
'XHhkMVx4OTgzXFxceDhmZilceDk2Ilx4MTZceDhkXHhhYlx4ZGFceDlkXHhiYUNH
XHhlM1x4YTRceDFiK0RceGNhfg=='
    app.run(debug=True)
```

functions.py

```
from os import path
from MySQLdb import connect
from MySQLdb import escape_string as sanitize
from flask import session
from passlib.hash import sha512_crypt
from itsdangerous import URLSafeSerializer
from itsdangerous import TimedSerializer
from werkzeug.utils import secure_filename

def processRegistration(form):
    username = sanitize(form.username.data)
    password = sha512_crypt.encrypt(sanitize(form.password.data))
    email = sanitize(form.email.data)
    fullname = sanitize(form.fullname.data)
    contact = (sanitize(form.contact.data)) or None
    reviewer = form.reviewer.data
    reviewer_choice = sanitize(str(form.reviewer_choice.data))

    cursor, dbconn = dbConnection()
    status = None;
```

```
    query = "SELECT * FROM users WHERE username = %s"
    query_result = cursor.execute(query, [username])

    if int(query_result) > 0:
        status = "Plz..Choose A Different Username..!!"
    else:
        try:
            query = "INSERT INTO users(username, password, fullname, email_id,
contact_no) VALUES(%s, %s, %s, %s, IFNULL(%s, DEFAULT(contact_no)));"
            query_result = cursor.execute(query, (username, password, fullname, email,
contact))
            if reviewer == True:
                query = "INSERT INTO reviewers(user_id, subject)
VALUES(LAST_INSERT_ID(), %s)"
                query_result = cursor.execute(query, [reviewer_choice])
            dbconn.commit()
            status = 0
        except:
            dbconn.rollback()
            status = "Oops.!! Something Went Wrong, Plz Try Again..!!"
    cursor.close()
    dbconn.close()
    return status

def processLogin(form):
    username = sanitize(form.username.data)
    password = sanitize(form.password.data)
    persistent = form.persistent.data

    status = False
    try:
```

```
        cursor, dbconn = dbConnection()
        query = "SELECT user_id, username, password FROM users WHERE username
= %s"
        query_result = cursor.execute(query, [username])
        query_result = cursor.fetchone()
        user_id = query_result[0]
        query_result = query_result[2]
    except:
        status = "No Such User Exists..!!"
    else:
        if sha512_crypt.verify(password, query_result):
            session['logged_in'] = True
            session['username'] = username
            if persistent:
                session.permanent = True
            status = True
        else:
            status = "Invalid Credentials..!!"
        query = "SELECT reviewer_id, subject from reviewers where user_id = %s"
        query_result = cursor.execute(query, [user_id])
        if int(query_result) > 0:
            query_result = cursor.fetchone()
            session['reviewer'] = query_result[0]
            session['subject'] = query_result[1]
    cursor.close()
    dbconn.close()
    return status

def processReset(form):
    username = sanitize(form.username.data)
    password = sanitize(form.password.data)
```

```
    email = None
    status = None
    token = None

    cursor, dbconn = dbConnection()
    query = "SELECT * FROM users WHERE username = %s"
    query_result = cursor.execute(query, [username])
    if int(query_result) <= 0:
        status = "No Such User Exists..!!"
    else:
        email = cursor.fetchone()[4]
        url_serial =
URLSafeSerializer("XHhkMVx4OTgzXFxceDhmZilceDk2Ilx4MTZceDhkXHhhYlx4ZG
FceDlkXHhiYUNHXHhlM1x4YTRceDFiK0RceGNhfg==")
        timed_serial =
TimedSerializer("XHhkMVx4OTgzXFxceDhmZilceDk2Ilx4MTZceDhkXHhhYlx4ZGFc
eDlkXHhiYUNHXHhlM1x4YTRceDFiK0RceGNhfg==")
        token = timed_serial.dumps(url_serial.dumps([username, password]))
    cursor.close()
    dbconn.close()
    return status, email, token

def resetVerifyCheck(key):
    url_serial =
URLSafeSerializer("XHhkMVx4OTgzXFxceDhmZilceDk2Ilx4MTZceDhkXHhhYlx4ZG
FceDlkXHhiYUNHXHhlM1x4YTRceDFiK0RceGNhfg==")
    timed_serial =
TimedSerializer("XHhkMVx4OTgzXFxceDhmZilceDk2Ilx4MTZceDhkXHhhYlx4ZGFc
eDlkXHhiYUNHXHhlM1x4YTRceDFiK0RceGNhfg==")
    status = None
    try:
```

```
        key = timed_serial.loads(key, max_age = 300)
        key = url_serial.loads(key)

        username = str(sanitize(key[0]))
        password = sha512_crypt.encrypt(sanitize(key[1]))
    except:
        status = "Oops, The Link Has Expired..!!"
        return status
    cursor, dbconn = dbConnection()
    query = "UPDATE users SET password = %s WHERE username = %s"
    try:
        cursor.execute(query, (password, username))
        dbconn.commit()
        status = "Password Updated Successfully..!!"
    except:
        dbconn.rollback()
        status = "Oops, Something Went Wrong Please Try Again..!!"
    cursor.close()
    dbconn.close()
    return status

def uploadArticle(form, file_obj):
    author = sanitize(form.author.data)
    email = sanitize(form.email.data)
    contact = sanitize(form.contact.data) or None
    address = sanitize(form.address.data)
    title = sanitize(form.title.data)
    article_subject = sanitize(str(form.article_subject.data))

    cursor, dbconn = dbConnection()
```

```
    query = "SELECT `AUTO_INCREMENT` FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = DATABASE()
AND TABLE_NAME = 'articles'";
    article_id = cursor.execute(query)
    article_id = cursor.fetchone()[0]

    status = uploadFiles(file_obj, article_id)
    if status == True:
        try:
            query = "INSERT INTO articles(title, subject, review_date) VALUES(%s, %s,
DEFAULT(review_date));"
            cursor.execute(query, (title, article_subject))

            query = "SELECT author_id FROM authors WHERE email_id = %s;"
            author_status = cursor.execute(query, [email])
            if int(author_status) > 0:
                author_id = cursor.fetchone()[0]
                query1 = "INSERT INTO article_author VALUES(%s, %s);"
                cursor.execute(query1, (article_id, author_id))
            else:
                query1 = "INSERT INTO authors(fullname, email_id, address, contact_no)
VALUES(%s, %s, %s, IFNULL(%s, DEFAULT(contact_no)));"
                query2 = "INSERT INTO article_author VALUES(%s,
LAST_INSERT_ID());"
                cursor.execute(query1,(author, email, address, contact))
                cursor.execute(query2, [article_id])
            dbconn.commit()
            status = "Article Uploaded Successfully"
        except:
            dbconn.rollback()
            status = "Oops, Something Went Wrong Please Try Again..!!"
```

```
    cursor.close()
    dbconn.close()
    return status

def uploadFiles(file_obj, filename):
    UPLOAD_FOLDER = 'articles'
    if 'article' not in file_obj:
        return "No File Attached..!"

    uploaded_file = file_obj['article']

    if uploaded_file.filename == '':
        return "No File Selected"

    try:
        if uploaded_file and allowedFiles(secure_filename(uploaded_file.filename)):
            uploaded_file.save(path.join(UPLOAD_FOLDER, str(filename)))
            return True
        return "Invalid File..!!"
    except:
        return "Oops, Something Went Wrong Please Try Again..!!"

def allowedFiles(filename):
    ALLOWED_EXTENSIONS = set(['txt'])
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

def listArticles():
    cursor, dbconn = dbConnection()
    query = "SELECT article_id, title, submit_date FROM articles;"
    query_result = cursor.execute(query)
```

```
    query_result = cursor.fetchall()
    length = len(query_result)
    excerpts = []
    for id in range(1, length+1):
        excerpts.append(getArticleExcerpt(id))
    cursor.close()
    dbconn.close()
    return query_result, excerpts

def getArticleExcerpt(filename):
    UPLOAD_FOLDER = "articles"
    try:
        filepath = path.join(UPLOAD_FOLDER, str(filename))
        article = open(filepath, 'r')
        article = article.readline()
        return article
    except:
        return "Article Not Found..!!"

def viewArticle(filename):
    UPLOAD_FOLDER = "articles"
    article = None
    status = True
    filename = sanitize(filename)
    cursor, dbconn = dbConnection()
    query = "SELECT articles.article_id, articles.title, articles.subject, articles.status,
articles.submit_date, articles.review_date, authors.fullname FROM articles INNER
JOIN article_author ON articles.article_id = article_author.article_id INNER JOIN
authors ON article_author.author_id = authors.author_id WHERE articles.article_id =
%s"
    article_details = cursor.execute(query, [filename])
```

```
    article_details = cursor.fetchone()
    try:
        filepath = path.join(UPLOAD_FOLDER, str(filename))
        article = open(filepath, 'r')
        article = article.read()
    except:
        status = "Oops, Something Went Wrong Please Try Again..!!"
    comment = viewComment(filename, cursor)
    cursor.close()
    dbconn.close()
    return article_details, article, status, comment


def reviewArticle(form, article_id):
    article_status = sanitize(str(form.status.data))
    article_id = sanitize(str(article_id))
    reviewer_id = session['reviewer']
    status = None
    cursor, dbconn = dbConnection()
    try:
        query = "UPDATE articles SET status = %s, review_date =
CURRENT_TIMESTAMP WHERE article_id = %s;"
        query1 = "INSERT INTO article_reviewer VALUES(%s, %s);"
        cursor.execute(query, (article_status, article_id))
        cursor.execute(query1, (article_id, reviewer_id))
        status = "Review Submitted Successfully..!!"
        dbconn.commit()
    except:
        dbconn.rollback()
        status = "Oops, Something Went Wrong Please Try Again..!!"
    cursor.close()
    dbconn.close()
```

```
    return status

def viewComment(filename, cursor):
    query = "SELECT users.fullname, comments.comment FROM comments INNER
JOIN reviewers ON reviewers.reviewer_id = comments.reviewer_id INNER JOIN
users ON reviewers.user_id = users.user_id WHERE article_id = %s;"
    query_result = cursor.execute(query, [filename])
    query_result = cursor.fetchall()
    return query_result

def commentArticle(form, article_id):
    status = "Thanks For Your Comment..!!"
    cursor, dbconn = dbConnection()
    article_id = sanitize(str(article_id))
    reviewer_id = session['reviewer']
    comment = sanitize(str(form.comment.data))
    try:
        if not (addComment(article_id, reviewer_id, comment, cursor, dbconn)):
            status = "You've Already Provided Your Comment..!!"
    except:
        dbconn.rollback()
        status = "Oops, Something Went Wrong Please Try Again..!!"
    return status

def addComment(article_id, reviewer_id, comment, cursor, dbconn):
    status = True
    query = "SELECT * FROM comments WHERE article_id = %s AND reviewer_id =
%s;"
    query_result = cursor.execute(query, (article_id, reviewer_id))
    if int(query_result) > 0:
        status = False
```

```
    else:
        try:
            query = "INSERT INTO comments(article_id, reviewer_id, comment)
VALUES(%s, %s, %s);"
            cursor.execute(query, (article_id, reviewer_id, comment))
            dbconn.commit()
        except:
            dbconn.rollback()
            status = False
    return status


def viewTopComments():
    cursor, dbconn = dbConnection()
    query = "SELECT users.fullname, comments.comment FROM comments INNER
JOIN reviewers ON reviewers.reviewer_id = comments.reviewer_id INNER JOIN
users ON reviewers.user_id = users.user_id LIMIT 7;"
    query_result = cursor.execute(query)
    query_result = cursor.fetchall()
    return query_result


def dbConnection():
    dbconn = connect(host='localhost', port=3306, user='root', passwd='password',
db='oprs')
    cursor = dbconn.cursor()
    return cursor, dbconn
```

forms.py

```
from wtforms import BooleanField
from flask_wtf import Form
from wtforms import PasswordField
from wtforms import SelectField
from wtforms import TextField
from wtforms import TextAreaField
from wtforms import SubmitField
from wtforms import validators
from flask_wtf.file import FileField

class RegistrationForm(Form):
    username = TextField('Username', [validators.Length(min=8, max=100),
validators.Required()], render_kw={"placeholder": "Username"})
    password = PasswordField('New Password', [validators.Length(min=8,
message='Passwords Must Be 8 Letters Or More..!!'), validators.Required(),
validators.EqualTo('confirm', message='Passwords Do Not Match..!!')],
render_kw={"placeholder": "Password"})
    confirm = PasswordField('Confirm Password', [validators.Required()],
render_kw={"placeholder": "Confirm Password"})
    email = TextField('E-mail Id', [validators.Required(),
validators.Email(message='Please Enter A Valid E-mail Address..!!')],
render_kw={"placeholder": "Email"})
    fullname = TextField('Full Name', [validators.Required(),
validators.Length(max=150)], render_kw={"placeholder": "Fullname"})
    contact = TextField('Contact Number', render_kw={"placeholder": "Contact
Number"})
    reviewer = BooleanField('Wanna Be A Reviewer..!!')
    reviewer_choice = SelectField('Select A Subject', choices = [('SYSTEM', 'SYSTEM'),
('WEB', 'WEB'), ('APPLICATION', 'APPLICATION')])
    tos = BooleanField('I Accept Terms Of Service!', [validators.required()])
```

```
    submit = SubmitField("Register")

class LoginForm(Form):
    username = TextField('Username', [validators.Length(min=8, max=100),
validators.Required()], render_kw={"placeholder": "Username"})
    password = PasswordField('Password', [validators.Length(min=8,
message='Passwords Must Be 8 Letters Or More..!!'),validators.Required()],
render_kw={"placeholder": "Password"})
    persistent = BooleanField("Remember Me..!!")
    submit = SubmitField("Login")

class PassReset(Form):
    username = TextField('Username', [validators.Length(min=8, max=100),
validators.Required()], render_kw={"placeholder": "Username"})
    password = PasswordField('New Password', [validators.Length(min=8,
message='Passwords Must Be 8 Letters Or More..!!'), validators.Required(),
validators.EqualTo('confirm', message='Passwords Do Not Match..!!')],
render_kw={"placeholder": "Password"})
    confirm = PasswordField('Confirm Password', [validators.Required()],
render_kw={"placeholder": "Confirm Password"})
    submit = SubmitField("Submit")

class UploadFile(Form):
    author = TextField('Author', [validators.Length(min=8, max=200),
validators.Required()], render_kw={"placeholder": "Author name"})
    email = TextField('E-mail Id', [validators.Required(),
validators.Email(message='Please Enter A Valid E-mail Address..!!')],
render_kw={"placeholder": "E-mail"})
    contact = TextField('Contact Number', [validators.Required()],
render_kw={"placeholder": "Contact Number"})
```

```
    address = TextField('Address', [validators.Required()], render_kw={"placeholder":
"Address"})
    title = TextField('Title', [validators.Length(max=255), validators.Required()],
render_kw = {"placeholder": "Title For Article"})
    article_subject = SelectField('Select A Subject', choices = [('SYSTEM', 'SYSTEM'),
('WEB', 'WEB'), ('APPLICATION', 'APPLICATION')])
    article = FileField()
    submit = SubmitField("Submit")

class ReviewForm(Form):
    status = SelectField('Update Article Status', choices = [('ACCEPTED',
'ACCEPTED'), ('REJECTED', 'REJECTED'), ('MODIFY', 'MODIFY')])
    submit = SubmitField("Review")

class CommentForm(Form):
    comment = TextAreaField('Comment', [validators.Required()])
    submit = SubmitField("Comment")
```

---

header.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>OPRS: Online Peer Review System</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" media="screen" href="/static/stylesheet.css">
        <link rel="stylesheet" media="screen" href="/static/form_styles.css">
        <link rel="shortcut icon" type="image/png" href="/static/favicon.ico"/>
        <link rel="stylesheet" href="/static/bootstrap.min.css"/>
        <script src="/static/jquery.min.js"></script>
        <script src="/static/bootstrap.min.js"></script>
```

```
        <script>
            $(document).ready(function () {
                $("#logoContainer").toggle().fadeIn(1500);
            });
        </script>
    </head>
    <script>
        function resize(){
            document.getElementsByTagName("body")[0].style["font-size"] =
document.body.clientWidth*(14/1280) + "px";
        }
    </script>
    <body onresize="resize()">
        <header class="container">
            <div class="row">
                <div class="col-md-12" id="logoContainer">
                    <a href="/">'PEER REVIEW!'</a>
                </div>
            </div>
            <div class="row">
                <div class="col-md-12" id="nav" style="">
                    <ul>
                        <li class="odd"><a href="/">Index</a></li>
                        <li class="even"><a href="/home/">Articles</a></li>
                        <li class="odd"><a href="/comments/">Comments</a></li>
                        <li class="even"><a href="#" data-toggle="modal" data-
target="#myModal">S[o]cial</a></li>
                        {% if 'logged_in' not in session: %}
                        <li class="odd"><a href="/login/">Sign-In</a></li>
                        {% else: %}
                        <li class="odd"><a href="/logout/">Sign-out</a></li>
```

```
                {% endif %}
            </ul>
        </div>
    </div>
</header>
<div class="modal fade" id="myModal" role="dialog">
    <div class="modal-dialog modal-sm">
        <div class="modal-content text-center">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal">&times;</button>
                <h4 class="modal-title" style="font-family:
'quicksandbold';">S[o]CIAL</h4>
            </div>
            <div class="modal-body">
                <img src="/static/fb.jpg"/>
                <br/>
                <img src="/static/twitter.png"/>
                <br/>
                <img src="/static/linkedin.png"/>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
<div class="container">
{% with messages = get_flashed_messages() %}
    {% if messages %}
```

```
            {% for message in messages %}
            <div class="row alert" style="margin-top: 2%;" role="alert">
                <div class="col-md-3"></div>
                <div class="col-md-6 alert-dismissible alert-warning fade in"
style="padding: 2.5%;">
                    <button type="button" class="close" data-dismiss="alert" aria-
label="Close">
                    <span aria-hidden="true">&times;</span></button>
                    <span>{{message}}</span>
                </div>
                <div class="col-md-3"></div>
            </div>
            {% endfor %}
        {% endif %}
    {% endwith %}
    </div>
    <br/><br/>
```

---

footer.html

```
    <footer class="container">
        <div class="row">
            <hr class="col-md-12 footerline" />
        </div>
        <div class="row">
            <div class="col-md-12 text-center footer">
                <p>All Rights Reserved :: OPRS Team</p>
                <p>Copyright &copy; <script>document.write(new
Date().getFullYear());</script></p>
            </div>
        </div>
    </footer>
```

```
    </body>
</html>
```

---

index.html

```
{% include "header.html" %}
        <div class="container home">
            <div class="row">
                <div class="col-md-8 hometitle">
                    <h3>What is Peer Review?</h3>
                </div>
                <div class="col-md-4"></div>
            </div>
            <div class="row">
                <div class="col-md-8 homedesc">
                    <p class="text-justify">
                        Peer review, known as refereeing in some academic fields, is a process of subjecting an author's scholarly work, research or ideas to the scrutiny of others who are experts in the same field. In journals peer review, the author's article is usually subjected to some initial checks to assess its suitability for review (for instance, incomplete articles or work that was patently pseudoscience would be declined without review), after which a small number of reviewers are selected. The task expected of the reviewers varies somewhat from journal to journal, but in essence, is usually to assist the journal's editor (who makes the final decision) on deciding whether or not to accept the article for publication. The reviewer will comment on the quality of the work done as well as on its originality and its importance.
                        <img src="/static/peer_review.png" alt="peer_review" style="width: 100%;"/>
                        Online peer review is emerging as the next step forward for many journal publishers in an ever increasing drive to take advantage of technological improvements in transferring data electronically over the internet. Traditionally the peer review process has been a paper-based system with authors submitting to a
```

journal using the postal service. Reviewers are then assigned, by a variety of methods, and sent copies of the manuscript to review and report upon. Technological advances have allowed other methods of transferring the information between parties such as facsimile and email, but although these methods speed up the process they suffer drawbacks in quality and size/platform limitations. More importantly they do not address how the information can be managed effectively.

```
            </p>
        </div>
        <div class="col-md-4">
            <a href="/uploads/"><img src="/static/submit.png" id="submit_img"
class="img-responsive"/></a>
            <div class="panelmain text-justify pull-right">
                <div class="panelhead" data-toggle="collapse" data-
target="#panelbody">
                    <span>LATEST REVIEWS</span>
                </div>
                <div class="panelbody collapse in" id="panelbody">
                    {% for comment in comments %}
                    <div class="reviews">
                        <p><b>{{ comment[0] }}: </b>{{ comment[1] }}</p>
                    </div>
                    {% endfor %}
                </div>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-md-8 hometitle">
            <h3>Is this just another Review System?</h3>
        </div>
        <div class="col-md-4"></div>
```

```
        </div>
        <div class="row">
          <div class="col-md-8 homedesc text-justify">
            <p>
              The purpose of this system is to provide article submission, peer review, document
              tracking, and semiautomatic correspondence with authors and reviewers. The main
              reason for developing such a system resides in the need for a simple review system
              that fulfill the needs of a much wider audience than the current systems and can be
              easily adapted to any future changes required. Another important reason for
              developing a new system over existing one's resides in the difficulty of managing and
              using such systems that have been built to cater the needs of their specific domains.
            </p>
            <p>
              Looking beyond the interests of the particular stakeholders, there are three main benefits advocated for peer review:
            </p>
            <p>1: Improvement in the quality of published papers.</p>
            <p>2: Filtering of the output of papers to the benefit of readers.</p>
            <p>3: A <i>‘seal of approval’</i> that the published work meets certain standards, in particular for lay readers.</p>
          </div>
          <div class="col-md-4"></div>
        </div>
      </div>
```

```
{% include "footer.html" %}
```

home.html

```
{% include "header.html" %}
{% from "__formhelper.html" import render_article %}
        <div class="container" id="main-content">
            {{ render_article(articles, excerpts, length) }}
        </div>
{% include "footer.html" %}
```

comments.html

```
{% include "header.html" %}
<div class="container">
    <div class="row">
        <div class="col-md-2"></div>
        <div class="col-md-8" id="comments_container">
        {% for comment in comments %}
            <div class="row" style="padding: 2%;">
                <div class="col-md-12">
                    <p>{{ comment[1] }}</p>
                    <p>By: <b>{{ comment[0] }}</b></p>
                </div>
            </div>
        {% endfor %}
        </div>
        <div class="col-md-2"></div>
    </div>
</div>
{% include "footer.html" %}
```

login.html

```
{% include "header.html" %}
{% from "__formhelper.html" import render_field %}
        <div class="row">
            <div class="col-md-1"></div>
            <div class="col-md-10">
                <div class="container1">
                    <section id="content1">
                        <form action="{{ url_for('login') }}" method="POST" name="form">
                            <h1 id="form-heading">Login Form</h1>
                            {{ render_field(form.username) }}
                            {{ render_field(form.password) }}
                            <label for="persistent">Remember
Me</label>{{ render_field(form.persistent) }}
                            {{ form.hidden_tag() }}
                            {{ form.submit }}
                            <a href="/reset/">Lost your password?</a>
                            <a href="/register/">Register</a>
                        </form>
                    </section>
                </div>
            </div>
            <div class="col-md-1"></div>
        </div>
{% include "footer.html" %}
```

register.html

```
{% include "header.html" %}
{% from "__formhelper.html" import render_field %}
    <div class="row">
      <div class="col-md-1"></div>
      <div class="col-md-10">
        <div class="container1">
          <section id="content1">
            <form action="{{ url_for('register') }}" method="POST" name="form">
              <h1 id="form-heading">Register</h1>
              {{ render_field(form.username) }}
              {{ render_field(form.password) }}
              {{ render_field(form.confirm) }}
              {{ render_field(form.email) }}
              {{ render_field(form.fullname) }}
              {{ render_field(form.contact) }}
              <label for="reviewer" style="margin: 2%;">Wanna Be A
Reviewer</label>{{ render_field(form.reviewer) }}
              <label for="reviewer_choice" style="margin: 2%;">Select Your
Field</label>{{ render_field(form.reviewer_choice) }}
              <label for="tos" style="margin: 2%;">I Accept The Terms &
Condition</label>{{ render_field(form.tos) }}
              {{ form.hidden_tag() }}
              {{ form.submit }}
            </form>
            <p>{{error}}</p>
          </section>
        </div>
      </div>
      <div class="col-md-1"></div>
    </div>
```

```
{% include "footer.html" %}
```

reset.html

```
{% include "header.html" %}
{% from "__formhelper.html" import render_field %}
    <div class="row">
        <div class="col-md-1"></div>
        <div class="col-md-10">
            <div class="container1">
                <section id="content1">
                    <form action="{{ url_for('passReset') }}" method="POST" name="form">
                        <h1 id="form-heading">Reset Form</h1>
                        {{ render_field(form.username) }}
                        {{ render_field(form.password) }}
                        {{ render_field(form.confirm) }}
                        {{ form.hidden_tag() }}
                        {{ form.submit }}
                    </form>
                </section>
            </div>
        </div>
        <div class="col-md-1"></div>
    </div>
{% include "footer.html" %}
```

uploads.html

```
{% include "header.html" %}
{% from "__formhelper.html" import render_field %}
      <div class="row">
         <div class="col-md-1"></div>
         <div class="col-md-10">
            <div class="container1">
               <section id="content1">
                  <form action="{{ url_for('uploads') }}" method="POST" name="form"
enctype=multipart/form-data>
                     <h1 id="form-heading">Upload Article</h1>
                     {{ render_field(form.author) }}
                     {{ render_field(form.email) }}
                     {{ render_field(form.contact) }}
                     {{ render_field(form.address) }}
                     {{ render_field(form.title) }}
                     <label for="article_subject" style="margin:
2%;">Subject</label>{{ render_field(form.article_subject) }}
                     <label for="article" style="margin: 2%;">Article File</label>
                     <div style="padding-left: 25%;">{{ render_field(form.article) }}</div>
                     {{ form.hidden_tag() }}
                     {{ form.submit }}
                  </form>
               </section>
            </div>
         </div>
         <div class="col-md-1"></div>
      </div>
{% include "footer.html" %}
```

view.html

```
{% include "header.html" %}
{% from "__formhelper.html" import render_field %}
{% from "__formhelper.html" import render_comments %}
    <div class="container">
      <div class="row">
        <div class="col-md-2"></div>
        <div class="col-md-8">
          <h3>{{ details[1] }}</h3>
          <h4>Subject: {{ details[2] }}</h4>
          <h6>Submitted By {{ details[6] }} On {{ details[4] }}</h6>
          <pre>{{ article }}</pre>
          <h6>Status: {{ details[3] }}</h6>
        </div>
        <div class="col-md-2"></div>
      </div>
      <div class="row">
        {% if 'reviewer' in session and session['subject'] == details[2] and details[3] !=
'ACCEPTED': %}
        <div class="col-md-2"></div>
        <div class="col-md-8">
          <h4>This article needs a review :</h4>
          <form action="/review/{{ details[0] }}" method="POST" name="form">
            {{ render_field(form.status) }}
            <br/>
            {{ form.hidden_tag() }}
            {{ form.submit }}
          </form>
        </div>
        {% endif %}
        <div class="col-md-2"></div>
```

```
        </div>
        <div class="row">
            <div class="col-md-2"></div>
            {% if 'reviewer' in session: %}
            <div class="col-md-8">
                <h4>Provide Your Valuable Comment :</h4>
                <form action="/comment/{{ details[0] }}" method="POST" name="form">
                    {{ form1.comment(cols="90", rows="10") | safe }}
                    <br/>
                    {{ form1.hidden_tag() }}
                    {{ form1.submit }}
                </form>
            </div>
            {% endif %}
            <div class="col-md-2"></div>
        </div>
        <br/><br/>
        {{ render_comments(comment) }}
    </div>
{% include "footer.html" %}
```

__formhelper.html

```
{% macro render_field(field) %}
        <div>
            {{ field(**kwargs) | safe }}
                {% if field.errors %}
                <ul class=errors>
                    {% for error in field.errors %}
                    <li>{{ error }}</li>
                    {% endfor %}
                </ul>
```

```
                {% endif %}
            </div>
{% endmacro %}

{% macro render_article(articles, excerpts, length) %}
        {% for id in range(0, length) %}
            <div class="row">
                <div class="col-md-2"></div>
                <div class="col-md-8 text-justify article">
                    <h3><a href="/view/{{ articles[id][0] }}">{{ articles[id][1] }}</a></h3>
                    <p>{{ excerpts[id] }}...<a href="/view/{{ articles[id]
[0] }}">[more]</a></h3></p>
                    <h6>{{ articles[id][2] }}</h6>
                    <a href="#main-content"><i>↑Back To Top</i></a>
                    <hr/>
                </div>
                <div class="col-md-2"></div>
            </div>
        {% endfor %}
{% endmacro %}

{% macro render_comments(comment) %}
        {% for c in comment %}
            <div class="row">
                <div class="col-md-3"></div>
                <div class="col-md-6 text-justify comments-main">
                    <br/>
                    <p>{{ c[1] }}</p>
                    <hr class="new-hr"/>
                    <h6><b>Commented By : {{ c[0] }}</b></h6>
                    <br/>
```

```
            </div>
            <div class="col-md-3"></div>
        </div>
    {% endfor %}
{% endmacro %}
```

---

stylesheet.css

```
* {
    padding: 0;
    margin: 0;
}

body{
    background: #f4f1e5 !important;
    overflow-x: hidden;
}

#logoContainer {
    margin: 1%;
    text-align: center;
}

#logoContainer a:hover{
    transition: ease-in .5s;
    text-shadow: 5px 5px 50px #A85543;
}

#logoContainer a{
    color: #544738;
    font-family: '3dumbregular', helvetica, sans-serif;
    font-size: 6em;
```

```
    text-decoration: none;
    transition: ease-out .5s;
}

#nav {
    text-align: center;
    font-family: '3dumbregular', helvetica, sans-serif;
    padding-top: .1%;
    padding-bottom: .1%;
    border-radius: 10%;
    border-top: 2px solid brown;
    border-bottom: 2px solid brown;
    width: inherit;
    margin-left: 18%;
    margin-right: 16%;
    margin-top: 1%;
}

#nav ul, li {
    margin: 0;
    padding: 0;
    overflow: hidden;
}

#nav ul li {
    float: left;
    list-style: none;
    text-align: center;
}

#nav li a {
```

```
    color: #544738;
    text-decoration: none;
    font-size: 30px;
    float: left;
    padding: 12px;
}

#nav li a:hover {
    color: #7eb9be;
}

.odd a:hover {
    -webkit-transition:All .5s ease-in-out;
    -moz-transition:All .5s ease-in-out;
    -o-transition:All .5s ease-in-out;
    -webkit-transform: rotate(-10deg) scale(1.2);
    -moz-transform: rotate(-10deg) scale(1.2);
    -o-transform: rotate(-10deg) scale(1.2);
}

.even a:hover {
    -webkit-transition:All .5s ease-in-out;
    -moz-transition:All .5s ease-in-out;
    -o-transition:All .5s ease-in-out;
    -webkit-transform: rotate(10deg) scale(1.2);
    -moz-transform: rotate(10deg) scale(1.2);
    -o-transform: rotate(10deg) scale(1.2);
}

.home{
    font-family: "quicksandregular";
```

```
}

.hometitle {
    font-family: "quicksandbold";
    border-radius: 10%;
    border-bottom: 2px solid brown;
    border-left: 2px solid brown;
}

.homedesc {
    padding: 2%;
    font-size: 18px;
}

.panelmain {
    width: 80%;
}

.panelhead {
    text-align: center;
    padding: 1%;
    color: white;
    font-family: "quicksandbold";
    font-size: 1.4em;
    border: 1px solid brown;
    border-radius: 15% 15% 0 0;
    background-color: brown;
    cursor: pointer;
}

.panelbody {
```

```
    border: 1px solid grey;
    border-radius: 0 0 5% 5%;
}

#submit_img {
    margin: 0 0 10% 30%;
}

.reviews {
    padding-top: 2%;
    padding-left: 5%;
    padding-right: 5%;
    border-top: 1px solid brown;
}

#main-content {
    font-family: "quicksandregular";
    font-size: 1.1em;
}

.article {
    background-color: #f8fee6;
    padding: 1% 1% 0 1%;
}

.article h3 {
    font-family: "quicksandbold";
    color: #3b3131;
}

.article h6 {
```

```
    font-family: "quicksandbold";
    color: #3b3131;
}

.article a {
    color: inherit;
}

.article hr {
    width: 80%;
    margin: 2% auto 0 auto;
    border-radius: 50%;
    border-bottom: 2px solid brown;
}

.comments-main {
    padding: 2%;
    margin-bottom: 2%;
    border-top: 1px solid grey;
    border-bottom: 1px solid grey;
    border-radius: 10%;
}

#comments_container {
    font-family: 'quicksandregular';
    background-color: #f8fee6;
    border: 1px solid ligthyellow;
    border-radius: 5%;
    padding: 2%;
}
```

```
.new-hr {
    width: 60%;
    margin: auto;
    border-radius: 50%;
    border-bottom: 2px solid brown;
}

.footerline {
    border: 1px solid brown;
    border-radius: 50%;
}

.footer {
    font-family: "quicksandregular";
    font-size: medium;
}

@font-face {
    font-family: '3dumbregular';
    src: url('3Dumb-webfont.eot');
    src: url('3Dumb-webfont.eot?#iefix') format('embedded-opentype'),
        url('3Dumb-webfont.woff') format('woff'),
        url('3Dumb-webfont.ttf') format('truetype'),
        url('3Dumb-webfont.svg#3dumbregular') format('svg');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: 'quicksandregular';
    src: url('quicksand-regular-webfont.woff2') format('woff2'),
```

```
        url('quicksand-regular-webfont.woff') format('woff');
    font-weight: normal;
    font-style: normal;

}

@font-face {
    font-family: 'quicksandbold';
    src: url('quicksand-bold-webfont.woff2') format('woff2'),
        url('quicksand-bold-webfont.woff') format('woff');
    font-weight: normal;
    font-style: normal;

}
```

---

form_styles.css

```
#form-heading{ font-size:28px; color:#563D64;}
small{ font-size:10px;}
b, strong{ font-weight:bold;}
#content1 a{ text-decoration: none; font-size: 12px;}
#content1 a:hover{ text-decoration: underline; }
.left { float:left; }
.right { float:right; }
.alignleft { float: left; margin-right: 15px; }
.alignright { float: right; margin-left: 15px; }
.clearfix:after,
form:after {
    content: ".";
    display: block;
    height: 0;
    clear: both;
```

```
    visibility: hidden;
}
.container1 { margin: 25px auto; position: relative; width: 900px; }
#content1 {
    background: #f4f1e5;
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f8f8f8',
endColorstr='#f9f9f9',GradientType=0 );
    -webkit-box-shadow: 0 1px 0 #fff inset;
    -moz-box-shadow: 0 1px 0 #fff inset;
    -ms-box-shadow: 0 1px 0 #fff inset;
    -o-box-shadow: 0 1px 0 #fff inset;
    box-shadow: 0 1px 0 #fff inset;
    border: 1px solid #c4c6ca;
    margin: 0 auto;
    padding: 25px 0 0;
    position: relative;
    text-align: center;
    text-shadow: 0 1px 0 #fff;
    width: 400px;
}
#content1 h1 {
    color: #7E7E7E;
    font: bold 25px Helvetica, Arial, sans-serif;
    letter-spacing: -0.05em;
    line-height: 20px;
    margin: 10px 0 30px;
}
#content1 h1:before,
#content1 h1:after {
    content: "";
    height: 1px;
```

```
    position: absolute;
    top: 10px;
    width: 27%;
}
#content1 h1:after {
    background: rgb(126,126,126);
    background: -moz-linear-gradient(left,  rgba(126,126,126,1) 0%, rgba(255,255,255,1)
100%);
    background: -webkit-linear-gradient(left,  rgba(126,126,126,1)
0%,rgba(255,255,255,1) 100%);
    background: -o-linear-gradient(left,  rgba(126,126,126,1) 0%,rgba(255,255,255,1)
100%);
    background: -ms-linear-gradient(left,  rgba(126,126,126,1) 0%,rgba(255,255,255,1)
100%);
    background: linear-gradient(left,  rgba(126,126,126,1) 0%,rgba(255,255,255,1)
100%);
    right: 0;
}
#content1 h1:before {
    background: rgb(126,126,126);
    background: -moz-linear-gradient(right,  rgba(126,126,126,1) 0%,
rgba(255,255,255,1) 100%);
    background: -webkit-linear-gradient(right,  rgba(126,126,126,1)
0%,rgba(255,255,255,1) 100%);
    background: -o-linear-gradient(right,  rgba(126,126,126,1) 0%,rgba(255,255,255,1)
100%);
    background: -ms-linear-gradient(right,  rgba(126,126,126,1) 0%,rgba(255,255,255,1)
100%);
    background: linear-gradient(right,  rgba(126,126,126,1) 0%,rgba(255,255,255,1)
100%);
    left: 0;
```

```
}
#content1:after,
#content1:before {
    background: #f9f9f9;
    background: -moz-linear-gradient(top,  rgba(248,248,248,1) 0%, rgba(249,249,249,1)
100%);
    background: -webkit-linear-gradient(top,  rgba(248,248,248,1)
0%,rgba(249,249,249,1) 100%);
    background: -o-linear-gradient(top,  rgba(248,248,248,1) 0%,rgba(249,249,249,1)
100%);
    background: -ms-linear-gradient(top,  rgba(248,248,248,1) 0%,rgba(249,249,249,1)
100%);
    background: linear-gradient(top,  rgba(248,248,248,1) 0%,rgba(249,249,249,1)
100%);
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f8f8f8',
endColorstr='#f9f9f9',GradientType=0 );
    border: 1px solid #c4c6ca;
    content: "";
    display: block;
    height: 100%;
    left: -1px;
    position: absolute;
    width: 100%;
}
#content1:after {
    -webkit-transform: rotate(2deg);
    -moz-transform: rotate(2deg);
    -ms-transform: rotate(2deg);
    -o-transform: rotate(2deg);
    transform: rotate(2deg);
    top: 0;
```

```
    z-index: -1;
}
#content1:before {
    -webkit-transform: rotate(-3deg);
    -moz-transform: rotate(-3deg);
    -ms-transform: rotate(-3deg);
    -o-transform: rotate(-3deg);
    transform: rotate(-3deg);
    top: 0;
    z-index: -2;
}
#content1 form { margin: 0 20px; position: relative }
#content1 form input[type="text"],
#content1 form input[type="password"] {
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
    -ms-border-radius: 3px;
    -o-border-radius: 3px;
    border-radius: 3px;
    -webkit-box-shadow: 0 1px 0 #fff, 0 -2px 5px rgba(0,0,0,0.08) inset;
    -moz-box-shadow: 0 1px 0 #fff, 0 -2px 5px rgba(0,0,0,0.08) inset;
    -ms-box-shadow: 0 1px 0 #fff, 0 -2px 5px rgba(0,0,0,0.08) inset;
    -o-box-shadow: 0 1px 0 #fff, 0 -2px 5px rgba(0,0,0,0.08) inset;
    box-shadow: 0 1px 0 #fff, 0 -2px 5px rgba(0,0,0,0.08) inset;
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -ms-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
    background: #eae7e7 url(https://cssdeck.com/uploads/media/items/8/8bcLQqF.png)
no-repeat;
```

```
    border: 1px solid #c8c8c8;
    color: #777;
    font: 13px Helvetica, Arial, sans-serif;
    margin: 0 0 10px;
    padding: 15px 10px 15px 40px;
    width: 80%;
}
#content1 form input[type="text"]:focus,
#content1 form input[type="password"]:focus {
    -webkit-box-shadow: 0 0 2px #ed1c24 inset;
    -moz-box-shadow: 0 0 2px #ed1c24 inset;
    -ms-box-shadow: 0 0 2px #ed1c24 inset;
    -o-box-shadow: 0 0 2px #ed1c24 inset;
    box-shadow: 0 0 2px #ed1c24 inset;
    background-color: #fff;
    border: 1px solid #ed1c24;
    outline: none;
}
#username { background-position: 10px 10px !important }
#password { background-position: 10px -53px !important }
#content1 form input[type="submit"] {
    background: rgb(254,231,154);
    background: -moz-linear-gradient(top,  rgba(254,231,154,1) 0%, rgba(254,193,81,1)
100%);
    background: -webkit-linear-gradient(top,  rgba(254,231,154,1)
0%,rgba(254,193,81,1) 100%);
    background: -o-linear-gradient(top,  rgba(254,231,154,1) 0%,rgba(254,193,81,1)
100%);
    background: -ms-linear-gradient(top,  rgba(254,231,154,1) 0%,rgba(254,193,81,1)
100%);
    background: linear-gradient(top,  rgba(254,231,154,1) 0%,rgba(254,193,81,1) 100%);
```

```
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#fee79a',
endColorstr='#fec151',GradientType=0 );
    -webkit-border-radius: 30px;
    -moz-border-radius: 30px;
    -ms-border-radius: 30px;
    -o-border-radius: 30px;
    border-radius: 30px;
    -webkit-box-shadow: 0 1px 0 rgba(255,255,255,0.8) inset;
    -moz-box-shadow: 0 1px 0 rgba(255,255,255,0.8) inset;
    -ms-box-shadow: 0 1px 0 rgba(255,255,255,0.8) inset;
    -o-box-shadow: 0 1px 0 rgba(255,255,255,0.8) inset;
    box-shadow: 0 1px 0 rgba(255,255,255,0.8) inset;
    border: 1px solid #D69E31;
    color: #85592e;
    cursor: pointer;
    font: bold 15px Helvetica, Arial, sans-serif;
    height: 35px;
    margin: 20px 0 35px 15px;
    position: relative;
    text-shadow: 0 1px 0 rgba(255,255,255,0.5);
    width: 120px;
}
#content1 form input[type="submit"]:hover {
    background: rgb(254,193,81);
    background: -moz-linear-gradient(top,  rgba(254,193,81,1) 0%, rgba(254,231,154,1)
100%);
    background: -webkit-linear-gradient(top,  rgba(254,193,81,1)
0%,rgba(254,231,154,1) 100%);
    background: -o-linear-gradient(top,  rgba(254,193,81,1) 0%,rgba(254,231,154,1)
100%);
```

```
    background: -ms-linear-gradient(top,  rgba(254,193,81,1) 0%,rgba(254,231,154,1)
100%);
    background: linear-gradient(top,  rgba(254,193,81,1) 0%,rgba(254,231,154,1) 100%);
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#fec151',
endColorstr='#fee79a',GradientType=0 );
}
#content1 form div a {
    color: #004a80;
    float: right;
    font-size: 12px;
    margin: 30px 15px 0 0;
    text-decoration: underline;
}
.button {
    background: rgb(247,249,250);
    background: -moz-linear-gradient(top,  rgba(247,249,250,1) 0%, rgba(240,240,240,1)
100%);
    background: -webkit-linear-gradient(top,  rgba(247,249,250,1)
0%,rgba(240,240,240,1) 100%);
    background: -o-linear-gradient(top,  rgba(247,249,250,1) 0%,rgba(240,240,240,1)
100%);
    background: -ms-linear-gradient(top,  rgba(247,249,250,1) 0%,rgba(240,240,240,1)
100%);
    background: linear-gradient(top,  rgba(247,249,250,1) 0%,rgba(240,240,240,1)
100%);
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f7f9fa',
endColorstr='#f0f0f0',GradientType=0 );
    -webkit-box-shadow: 0 1px 2px rgba(0,0,0,0.1) inset;
    -moz-box-shadow: 0 1px 2px rgba(0,0,0,0.1) inset;
    -ms-box-shadow: 0 1px 2px rgba(0,0,0,0.1) inset;
    -o-box-shadow: 0 1px 2px rgba(0,0,0,0.1) inset;
```

```
    box-shadow: 0 1px 2px rgba(0,0,0,0.1) inset;
    -webkit-border-radius: 0 0 5px 5px;
    -moz-border-radius: 0 0 5px 5px;
    -o-border-radius: 0 0 5px 5px;
    -ms-border-radius: 0 0 5px 5px;
    border-radius: 0 0 5px 5px;
    border-top: 1px solid #CFD5D9;
    padding: 15px 0;
}
.button a {
    background: url(https://cssdeck.com/uploads/media/items/8/8bcLQqF.png) 0 -112px
no-repeat;
    color: #7E7E7E;
    font-size: 17px;
    padding: 2px 0 2px 40px;
    text-decoration: none;
    -webkit-transition: all 0.3s ease;
    -moz-transition: all 0.3s ease;
    -ms-transition: all 0.3s ease;
    -o-transition: all 0.3s ease;
    transition: all 0.3s ease;
}
.button a:hover {
    background-position: 0 -135px;
    color: #00aeef;
}
```

db_structure.sql

```
-- Host: localhost    Database: oprs
-- ------------------------------------------------------
-- Server version    10.1.23-MariaDB-9+deb9u1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `article_author`
--

DROP TABLE IF EXISTS `article_author`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `article_author` (
  `article_id` int(11) NOT NULL,
```

```
  `author_id` int(11) NOT NULL,
  KEY `fk` (`article_id`),
  KEY `fk1` (`author_id`),
  CONSTRAINT `article_author_ibfk_1` FOREIGN KEY (`article_id`) REFERENCES
`articles` (`article_id`),
  CONSTRAINT `article_author_ibfk_2` FOREIGN KEY (`author_id`) REFERENCES
`authors` (`author_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Table structure for table `article_reviewer`
--

DROP TABLE IF EXISTS `article_reviewer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `article_reviewer` (
  `article_id` int(11) NOT NULL,
  `reviewer_id` int(11) NOT NULL,
  KEY `fk` (`article_id`),
  KEY `fk1` (`reviewer_id`),
  CONSTRAINT `article_reviewer_ibfk_1` FOREIGN KEY (`article_id`)
REFERENCES `articles` (`article_id`) ON DELETE CASCADE ON UPDATE
CASCADE,
  CONSTRAINT `article_reviewer_ibfk_2` FOREIGN KEY (`reviewer_id`)
REFERENCES `reviewers` (`reviewer_id`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
-- Table structure for table `articles`
--

DROP TABLE IF EXISTS `articles`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `articles` (
  `article_id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL,
  `subject` enum('SYSTEM','WEB','APPLICATION') NOT NULL,
  `status` enum('ACCEPTED','REJECTED','PENDING','MODIFY') NOT NULL
DEFAULT 'PENDING',
  `submit_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `review_date` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`article_id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `authors`
--

DROP TABLE IF EXISTS `authors`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `authors` (
  `author_id` int(11) NOT NULL AUTO_INCREMENT,
  `fullname` varchar(200) NOT NULL,
  `email_id` varchar(255) NOT NULL,
  `address` text NOT NULL,
```

```
  `contact_no` varchar(15) NOT NULL DEFAULT '0000000000',
  PRIMARY KEY (`author_id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `comments`
--

DROP TABLE IF EXISTS `comments`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `comments` (
  `comment_id` int(11) NOT NULL AUTO_INCREMENT,
  `article_id` int(11) NOT NULL,
  `reviewer_id` int(11) NOT NULL,
  `comment` text NOT NULL,
  PRIMARY KEY (`comment_id`),
  KEY `fk` (`article_id`),
  KEY `fk1` (`reviewer_id`),
  CONSTRAINT `comments_ibfk_1` FOREIGN KEY (`article_id`) REFERENCES
`articles` (`article_id`),
  CONSTRAINT `comments_ibfk_2` FOREIGN KEY (`reviewer_id`) REFERENCES
`reviewers` (`reviewer_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `reviewers`
--
```

```
DROP TABLE IF EXISTS `reviewers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `reviewers` (
  `reviewer_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `subject` enum('SYSTEM','WEB','APPLICATION') NOT NULL,
  PRIMARY KEY (`reviewer_id`),
  KEY `fk` (`user_id`),
  CONSTRAINT `reviewers_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users`
(`user_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(100) NOT NULL,
  `password` binary(120) DEFAULT NULL,
  `fullname` varchar(200) NOT NULL,
  `email_id` varchar(255) NOT NULL,
  `contact_no` varchar(15) NOT NULL DEFAULT '0000000000',
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS
*/;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

5.2.2 CODE EFFICIENCY

This whole project is developed with efficiency being the top priority. The code is highly optimised and efficient. Some of the methods of code optimization utilised while implementing the project are as follows:

Dynamic Generation of User Interface: To prevent re-writing of the same code segment again and again multiple methodologies are used.

- Forms to take input from users are designed using '*Classes*' based on object-oriented principles so that they can be utilised again without re-writing the whole code.
- Displaying of these forms are done using '*Macros*' so that the same *HTML* code should not be repeated.

Database Connection: Connection definition to the database is also wrapped in a single method which can be called by any other method without the need to implement it's own connection. In addition to this, the library used for database connectivity offers connection pooling which maintains a cache of multiple connections so that the connections can be reused when future requests to the database are made.

Database Queries: 99.8% of all database queries are of O(1) or '*constant-running time*'. These queries are designed in such a way so as to avoid unnecessary load on the database server.

Code and Database Consistency: The code of the project is written following the '*mixedCase*' variable standard. Proper comments are provided wherever necessary. Multiple rollback conditions are used to maintain database consistency.

Modular Code: The complete system has been developed using modular approach. Each feature is encapsulated into it's own function and classes are used to organise objects whose properties are constantly being invoked. Nested functions are completely avoided and the size of functions are kept to a minimum for the ease of code reading.

5.3 TESTING APPROACH

The development process for the '*Online Peer Review System*' have been closely integrated with testing mechanism. A standard model of testing was followed and the system was tested with '*Functional Approach*'.

In functional approach, the software program or system under test is viewed as a '*black box*". The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test.

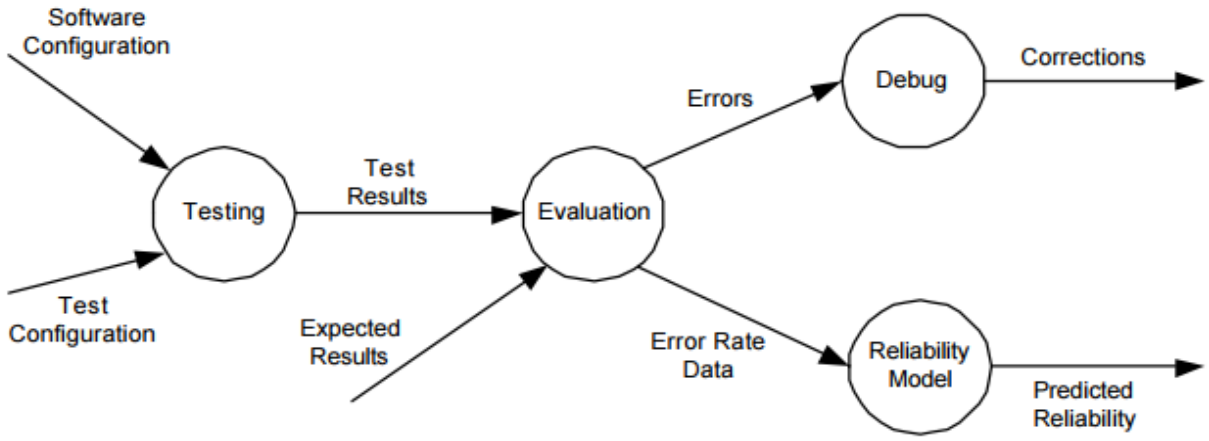


Figure 10 : Testing Process

Examples of expected results, some times are called '*test oracles*', include requirement/design specifications, hand calculated values, and simulated results. Functional testing emphasizes on the external behavior of the software entity.

5.4 MODIFICATIONS AND IMPROVEMENTS

After implementation of the project, some short-comings were bound to come to light. Many bugs and limitations of the project were found during the implementation and testing phase. A brief description of the bugs and shortcomings is as follows:

Secure Password-Reset Policy: After implementation of the project, it was found that users may require a way to securely change their authentication passwords. This feature was implemented in this phase.

Top Comments Feature: Another feature which was implemented after the original development cycle. This feature was added to display top comments across all articles in new page and also in widget format on the homepage sidebar.

6 RESULTS AND DISCUSSION

6.1 TEST REPORTS

TC01 – Login Page : Authenticate successfully on *Peer Review* website.

Correct Input Parameters: username – oprs and password – peer_review

- **Status:** Successful.

Incorrect Input Parameters: username – abcd and password – xyz

- **Status:** Failed.

Execution Duration: Less than 1 seconds.

==========

TC02 – Registration Page : Register successfully on *Peer Review* website.

Correct Input Parameters: username – oprs, password – peer_review, email – oprs@gmail.com, fullname – peer review system

- **Status:** Successful.

Incorrect Input Parameters: username – abcd and password – xyz, email – abcd@xyz.com

- **Status:** Failed.

Execution Duration: Less than 1.5 seconds.

==========

TC03 – Upload Article : Upload an article successfully on *Peer Review* website.

Correct Input Parameters: authorname – oprs, title – peer_review, subject – web and file - 'article.txt'

- **Status:** Successful.

Incorrect Input Parameters: authorname – oprs, title – peer_review, subject – web and file - 'article.exe'

- **Status:** Failed.

Execution Duration: Less than 2 seconds.

==========

TC04 – Comment Form : Provide a comment successfully on *Peer Review* website.

Correct Input Parameters: comment – 'Great article!'

- **Status:** Successful.

Incorrect Input Parameters: comment – "<script>alert('invalid');</script>"

- **Status:** Failed.

Execution Duration: Less than 1 seconds.

==========

TC05 – Review Form : Provide a review successfully on *Peer Review* website.

Correct Input Parameters: review – 'System', 'Web' or 'Application'

- **Status:** Successful.

Incorrect Input Parameters: review – "xyz"

- **Status:** Failed.

Execution Duration: Less than 1 seconds.

==========

6.2 USER DOCUMENTATION (SCREENSHOTS)

INDEX PAGE

'PEER REVIEW!'

Index Articles Comments S[o]cial Sign-In

What is Peer Review?

Peer review, known as refereeing in some academic fields, is a process of subjecting an author's scholarly work, research or ideas to the scrutiny of others who are experts in the same field. In journals peer review, the author's article is usually subjected to some initial checks to assess its suitability for review (for instance, incomplete articles or work that was patently pseudoscience would be declined without review), after which a small number of reviewers are selected. The task expected of the reviewers varies somewhat from journal to journal, but in essence, is usually to assist the journal's editor (who makes the final decision) on deciding whether or not to accept the article for publication. The reviewer will comment on the quality of the work done as well as on its originality and its importance.

Online peer review is emerging as the next step forward for many journal publishers in an ever increasing drive to take advantage of technological improvements in transferring data electronically over the internet. Traditionally the peer review process has been a paper-based system with authors submitting to a journal using the postal service. Reviewers are then assigned, by a variety of methods, and sent copies of the manuscript to review and report upon. Technological advances have allowed other methods of transferring the information between parties such as facsimile and email, but although these methods speed up the process they suffer drawbacks in quality and size/platform limitations. More importantly they do not address how the information can be managed effectively.

LATEST REVIEWS

Aditya Pratap Singh: Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Abhishek Pratap Singh: Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Abhishek Pratap Singh: Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Raghvendra Nath Tripathi: Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Aditya Pratap Singh: Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Is this just another Review System?

The purpose of this system is to provide article submission, peer review, document tracking, and semiautomatic correspondence with authors and reviewers. The main reason for developing such a system resides in the need for a simple review system that fulfill the needs of a much wider audience than the current systems and can be easily adapted to any future changes required. Another important reason for developing a new system over existing one's resides in the difficulty of managing and using such systems that have been built to cater the needs of their specific domains.

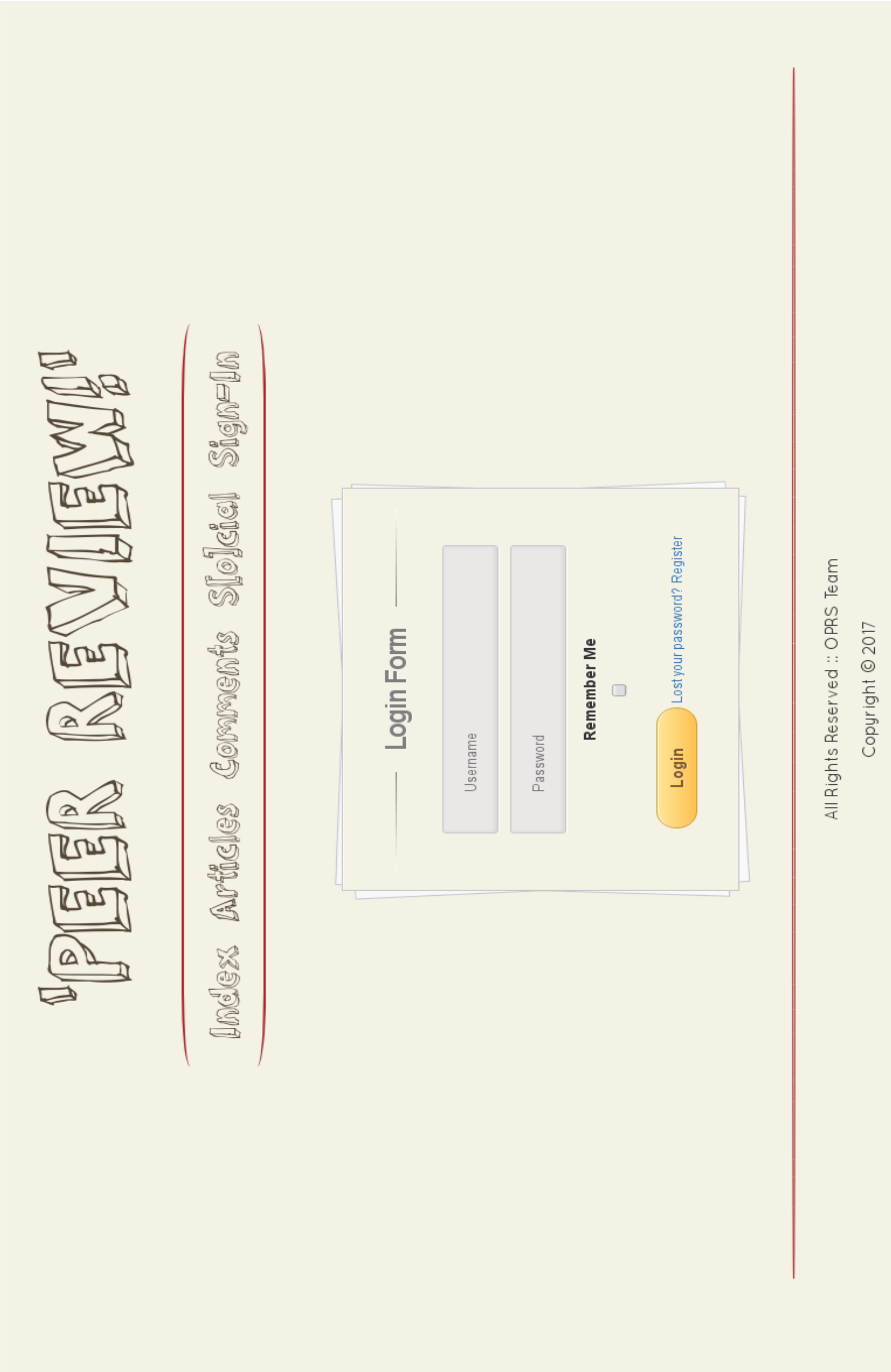
Looking beyond the interests of the particular stakeholders, there are three main benefits advocated for peer review:

1: Improvement in the quality of published papers.

2: Filtering of the output of papers to the benefit of readers.

3: A *'seal of approval'* that the published work meets certain standards, in particular for lay readers.

LOGIN PAGE


PEER REVIEW!
Index
Articles
Comments
Sign-In
Login Form
Username
Password
Remember Me
Lost your password? Register
Login
All Rights Reserved :: OPRS Team
Copyright © 2017

REGISTRATION PAGE

PASSWORD RESET PAGE

HOME PAGE


'PEER REVIEW!'
Index Articles Comments S[o]cial Sign-out
Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming recusabo id nec.
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. ...[more]
2017-07-08 23:23:00
↑Back To Top
Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming recusabo id nec.
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. ...[more]
2017-07-08 23:26:12
↑Back To Top
Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming recusabo id nec.
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. ...[more]
2017-07-08 23:27:25
↑Back To Top
Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming recusabo id nec.
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. ...[more]
2017-07-08 23:37:41
↑Back To Top
Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming recusabo id nec.
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. ...[more]
2017-07-08 23:38:30
↑Back To Top
All Rights Reserved :: OPRS Team
Copyright © 2017

ARTICLE PAGE

'PEER REVIEW!'

Index Articles Comments S[o]cial Sign-out

Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming recusabo id nec.

Subject: SYSTEM

Submitted By Aditya Pratap Singh On 2017-07-08 23:23:00

```
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec.
Nisl mazim no sit, cum ex veri volutpat postulant.
Cu erant albucius abhorreant has. Quo gloriatur inciderint te.
Et mea mentitum officiis repudiandae, per soluta graeco efficiantur eu,
mei wisi sensibus ad.
Ut tation legendos disputando sit, eu facilis petentium definiebas nam.

Paulo definitionem vim et, simul doctus te qui.
Ut eum esse nonumes, modus sadipscing ne pri,
in esse gubergren laboramus his.
Has in maiorum posidonium. Elit diceret lobortis ut cum.
Ferri assueverit sadipscing ad ius, pri homero malorum et,
at praesent repudiare efficiantur sea.
Agam civibus cotidieque an ius.

Pro et sumo consequat scriptorem, eruditi atomorum te eos.
Cu efficiendi intellegam delicatissimi quo,
quo ea dicant verear quaestio.
His commune mandamus ex, euripidis hendrerit has eu,
persius eripuit eu vel.
Modus labitur has et, ei referrentur necessitatibus mel.
Deserunt disputando eloquentiam mel ne.

Sonet recusabo et duo. Ex sit fuisset singulis,
no ius omittam lobortis adversarium.
Veri quaeque ea pri, mei menandri mediocritatem an.
Te usu quis indoctum definiebas, cum an erant viris saepe.
Ut populo tamquam tincidunt eum, cu sea dicant consequat.
```

Status: PENDING

This article needs a review :

ACCEPTED ▾

Review

Provide Your Valuable Comment :

Comment

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Commented By : Abhishek Pratap Singh

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

Commented By : Raghvendra Nath Tripathi

'PEER REVIEW!'

Index Articles Comments S[o]cial Sign-In

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

By: **Aditya Pratap Singh**

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

By: **Abhishek Pratap Singh**

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

By: **Abhishek Pratap Singh**

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

By: **Raghvendra Nath Tripathi**

Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec. Nisl mazim no sit, cum ex veri volutpat postulant.

By: **Aditya Pratap Singh**

SOCIAL PAGE

Index
Signout
S[o]CIAL
Like
Follow
Linked in
Close
Lorem ipsum dolor sit amet, est adipisci honestatis cu, doming
recusabo id nec.
Subject: SYSTEM
Submitted By Aditya Pratap Singh On 2017-07-08 23:23:00
Lorem ipsum dolor sit amet, est adipisci honestatis cu, mazim no doming recusabo id nec.
Nisl mazim no sit, cum ex veri volutpat postulant.
Cu erant albucius abhorreant has. Quo gloriatur incideriint te.
Et mea mentitum officiis repudiandae, per soluta graeco efficiantur eu,
mei wisi sensibus ad.
Ut tation legendos disputando sit, eu facilis petentium definiebas nam.
Paulo definitionem vim et, simul doctus te qui.
Ut eum esse nonumes, modus sadipscing ne pri,
in esse gubergren laboramus his.
Has in maiorum posidonium. Elit diceret lobortis ut cum.
Ferri assueverit sadipscing ad ius, pri homero malorum et,
at praesent repudiare efficiantur sea.
Agam civibus cotidieque an ius.
Pro et sumo consequat scriptorem, eruditi atomorum te eos.
Cu efficiendi intellegam delicatissimi quo,
quo ea dicant verear quaestio.
His commune mandamus ex, euripidis hendrerit has eu,
persius eripuit eu vel.
Modus labitur has et, ei referrentur necessitatibus mel.
Deserunt disputando eloquentiam mel ne.
Sonet recusabo et duo. Ex sit fuisset singulis,
no ius omittam lobortis adversarium.
Veri quaeque ea pri, mei menandri mediocritatem an.
Te usu quis indoctum definiebas, cum an erant viris saepe.
Ut populo tamquam tincidunt eum, cu sea dicant consequat.

7 CONCLUSIONS

7.1 CONCLUSION

The evaluation of authors and current working practices have shown the willingness of authors and users to adopt electronic submission and peer review. An important step in the development of the '*Online Peer Review System*' has been the implementation of a completely new user interface, specifically designed for the group of people who are used to the manual idea of peer review.

The key requirements of the publisher, the authors and the general user have been implemented in the new system in simple intuitive manner. The system provides an incentive for authors to submit as they can view their submissions and check the progress of the submission throughout the entire process.

The modularity of the system will allow additional features to be incorporated as authors and users adopt electronic workflow. Perhaps more importantly the technical considerations in the system implementation will give publishers, universities the option of hosting the software themselves and integrating it with their existing article or manuscripts review systems.

7.2 LIMITATIONS OF THE SYSTEM

After the development process, in addition to previous limitations found during the analysis phase, many new one's were also found. Some of these limitations which had high priority were resolved and some low priority one's were left to be resolved during the next version of the system because of time constraints. A brief description of these limitations are:

- **Multiple File Formats:** Because of time constraints, currently the system only supports '*txt*' format files for uploading articles. This limitation will be further removed with support to multiple formats like pdf, doc, rtf, odt etc.
- **UX/UI Improvement:** User interface and the overall user experience of the system can be greatly improved in the future versions. A better layout of the

user interface could be created which was not possible in this iteration of the development cycle because of a very tight schedule.

- **Database Improvement:** Database response time can be further improved by implementing highly optimised '*indexes*' on tables with large of amounts of data. While the database was properly '*normalized*' before the development process, '*views*' were not implemented. In the next development cycle optimised views can be designed on tables which are used frequently to improve the overall response time.

7.3 FUTURE SCOPE OF THE PROJECT

Upon completion of the project, many new areas of development have been found which can be further investigated or worked-upon. Features or functionalities which could not be implemented due to time constraints are as follows:

- **Subscription Service:** This feature would allow a user to opt for an e-mail subscription. An e-mail would be sent to the subscribed user whenever a new article would be published on topic of the user's choice.
- **Article Ratings:** In addition to the comments feature, this feature will provide the users an option to rate an article '*star ratings*' based on which the top articles will be displayed in a seperate module.
- **Social Media Connectivity:** In future versions, the '*Online Peer Review System*' could be connected with social media platforms like '*Facebook, Twitter, Google Plus*' etc to authenticate a user. The user would also get an option to share an article right from the '*Peer Review*' website.

8 REFERENCES

1. Peer Review: benfits, perceptions and alternatives by Mark Ware – Publishing Research Consortium, 2008, London.
2. Survey of Web-Technologies [White Paper] by Barry Doyle and Cristina Videira Lopes – University of California, Irvine.
3. https://en.wikipedia.org/wiki/Python_(programming_language).
4. The Development of an Online Submission and Peer Review System [Research Paper] by Paul Pavey, Steve Probets, David Brailsford – University of Nottingham, England.
5. Designing with Web Standards by Jeffery Zeldman, New Riders Publishing – 2003 Indianapolis.
6. Integrating object technology and the Web – 1997 W3C http://www.w3.org/OOP/
7. Python: Language of choice for EAI by A. Trauring – EAI Journal Publishing 2003.
8. Scripting: Higher-level programming for the 21st century by John K. Ousterhout – Sun Microsystems Laboratories 1998.
9. Object Oriented Web Application Development by Hans-W. Gellersen and Martin Gaedke – University of Karlsruhe 1999.
10. Tools and Approaches for Developing Data-Intensive Web Applications by Piero Fraternali - ACM Computing Surveys 1999.
11. Process Models in Software Engineering by Walt Scacchi – Institute for Software Research, University of California, Irvine 2001.
12. Software Testing Techniques – Technology Maturation and Research Strategy by Lu Luo – Carnegie Mellon University.
13. The Elements of User Interface Design by John Wiley & Sons, 1997.
14. A Peer Review System to Enhance Collaborative Learning by Brandon Holt, Luke Komiskey, Joline Morrison, and Mike Morrison.