

# *CONFERENCE MANAGEMENT SYSTEM*

*MINI PROJECT*

*SUBMITTED BY*

MOHMED IMTHIAZ I

110122104071

ARUNACHALAM A

110122104020

MOHAMED ASLAM M

110122104048

MOHAMED ARSHATH A R

1101221040305

*in partial fulfillment of the award of the  
degree of*

*BACHELOR OF ENGINEERING*

*IN*

*COMPUTER SCIENCE ENGINEERING*



*AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING*

*ANNA UNIVERSITY :CHENNAI 600 055*

*MAY-2025*

# **ANNA UNIVERSITY: CHENNAI – 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this mini project report “**STUDENT INFORMATION SYSTEM**” is the bonafide work of **MOHAMED IMTHIAZ I (110122104071) , ARUNACHALAM A (110122104020), MOHAMED ASLAM M (110122104048), MOHAMED ARSHATH A R (1101221040305)** who carried out project work under my supervision

### **SIGNATURE**

**Dr. ARIF ABDUL RAHUMAN S  
HEAD OF THE DEPARTMENT  
PROFESSOR**

Department of Computer Science  
and Engineering,  
Aalim Muhammed Salegh,  
College of Engineering  
Muthapudupet, Avadi IAF,  
Chennai 600 055.

### **SIGNATURE**

**Mr. PASUPATHI M  
ASSISTANT PROFESSOR**

Department of Computer Science  
and Engineering,  
Aalim Muhammed Salegh,  
College of Engineering,  
Muthapudupet, Avadi IAF,  
Chennai 600 055.

Submitted for viva voce held on..... at Aalim  
Muhammad Salegh

College of Engineering

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **ABSTRACT**

The Conference Management System project aims to streamline the process of managing conference submissions, reviews, and registrations through an automated platform. This system facilitates the submission of papers by candidates, their review by designated reviewers, and the subsequent acknowledgment sent to the candidates. Key functionalities of the system include user authentication, paper submission, reviewer assignment, acknowledgment generation, candidate selection, and registration management. Candidates can easily submit their papers online, which are then reviewed by assigned reviewers. Based on the review, acknowledgements are sent to candidates, and selected candidates proceed with registration. The software interface is developed using Rational Rose Software, providing a user-friendly platform for candidates, reviewers, and administrators. The backend infrastructure utilizes a database to store user details, paper submissions, review outcomes, and registration information. UML diagrams, including use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams, deployment diagrams, and package diagrams, are used to model various aspects of the system architecture and functionality. These diagrams provide a comprehensive understanding of the system's structure, interactions, and workflow. The Conference Management System project successfully implements an automated platform for managing conference submissions, reviews, and registrations, enhancing efficiency and convenience for all stakeholders involved.

## ❖ INTRODUCTION:

CASE tools known as Computer-aided software engineering tools is a kind of component-based development which allows its users to rapidly develop information systems. The main goal of case technology is the automation of the entire information systems development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation. Component based manufacturing has several advantages over custom development. The main advantages are the availability of high quality, defect free products at low cost and at a faster time. The prefabricated components are customized as per the requirements of the customers. The components used are pre-built, ready-tested and add value and differentiation by rapid customization to the targeted customers. However the products we get from case tools are only a skeleton of the final product required and a lot of programming must be done by hand to get a fully finished, good product.

## ❖ CHARACTERISTICS OF CASE:

Some of the characteristics of case tools that make it better than customized development are;

- ☞ It is a graphic oriented tool.
- ☞ It supports decomposition of process.

Some typical CASE tools are:

- ☞ Unified Modeling Language
- ☞ Data modeling tools
- ☞ Source code generation tools

## ❖ INTRODUCTION TO UML (UNIFIED MODELING LANGUAGE):

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modeling systems, ranging from enterprise information system to distributed web based application and even to real time embedded system. It is a very expensive language addressing all views needed to develop and then to display system even though understand to use. Learning to apply UML effectively starts forming a conceptual mode of languages which requires learning.

Three major language elements:

- ☞ UML basic building blocks

- ☞ Rules that dictate how this building blocks put together .

- ☞ Some common mechanism that apply throughout the language

The primary goals in the design of UML are:

1. Provides users ready to use, expressive visual modeling language as well so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies. Every complex system

is best approached through a small set of nearly independent views of a model. Every model can be expressed at different levels of fidelity. The best models are connected to reality. The UML defines nine graphical diagrams:

1. Class diagram
2. Use-case diagram
3. Behavior diagram
  - 3.1. Interaction diagram
    - 3.1.1. sequence diagram
    - 3.1.2. collaboration diagram
  - 3.2. state chart diagram
  - 3.3. activity diagram
4. Implementation diagram
  - 4.1 component diagram

### **1. UML class diagram:**

The UML class diagram is also known as object modeling. It is a static analysis diagram. These diagrams show the static structure of the model. A class diagram is a connection of static model elements, such as classes and their relationships, connected as a graph to each other and to their contents.

### **2. Use-case diagram:**

The functionality of a system can be described in a number of different use-cases, each of which represents a specific flow of events in a system. It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases.

### **3. Behavior diagram:**

It is a dynamic model unlike all the others mentioned before. The objects of an object oriented system are not static and are not easily understood by static diagrams. The behavior of the class's instance (an object) is represented in this diagram. Every use-case of the system has an associated behavior diagram that indicates the behavior of the object. In conjunction with the use-case diagram we may provide a script or interaction diagram to show a time line of events. It consists of sequence and collaboration diagrams.

### **4. Interaction diagram:**

It is the combination of sequence and collaboration diagram. It is used to depict the flow of events in the system over a timeline. The interaction diagram is a dynamic model which shows how the system behaves during dynamic execution.

### **5. State chart diagram:**

It consists of state, events and activities. State diagrams are a familiar technique to describe the behavior of a system. They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object. In most OO techniques, state diagrams are drawn for a single class to show the lifetime behavior of a single object.

### **6. Activity diagram:**

It shows organization and their dependence among the set of components. These diagrams are particularly useful in connection with workflow and in describing behavior that has a lot of parallel processing. An activity is a state of doing something: either a real-world process, or the execution of a software routine.

### **7. Implementation diagram:**

It shows the implementation phase of the systems development, such as the source code structure and the run-time implementation structure. These are relatively simple high level diagrams compared to the others seen so far. They are of two sub-diagrams, the component diagram and the deployment diagram.

## **8. Component diagram:**

These are organizational parts of a UML model. These are boxes to which a model can be decomposed. They show the structure of the code itself. They model the physical components such as source code, user interface in a design. It is similar to the concept of packages.

## **9. Deployment diagram:**

The deployment diagram shows the structure of the runtime system. It shows the configuration of runtime processing elements and the software components that live in them. They are usually used in conjunction with deployment diagrams to show how physical modules of code are distributed on the system.

### **◆ NOTATION ELEMENTS:**

These are explanatory parts of UML model. They are boxes which may apply to describe and remark about any element in the model. They provide the information for understanding the necessary details of the diagrams.

### **◆ Relations in the UML:**

These are four kinds of relationships used in an UML diagram, they are:

- ☐ Dependency
- ☐ Association
- ☐ Generalization
- ☐ Realization

### **Dependency:**

It is a semantic relationship between two things in which a change one thing affects the semantics of other things. Graphically a dependency is represented by a non-continuous line.



**Association:**

It is a structural relationship that describes asset of links. A link is being connected among objects. Graphically association is represented as a solid line possibly including label.

**Generalization:**

It is a specialized relationship in which the specialized elements are substitutable for object of the generalized element. Graphically it is a solid line with hollow arrow head parent.

**Realization:**

It is a semantic relation between classifiers. Graphically it is represented as a cross between generalization and dependency relationship.

**Where UML can be used:**

UML is not limited to modeling software. In fact it is expressive to model non-software such as to show in structure and behavior of health case system and to design the hardware of the system.

**Conceptual model be UML:**

UML you need to form the conceptual model of UML. This requires three major elements:

- ☞ UML basic building blocks.
- ☞ Rules that dictate how this building blocks are put together.
- ☞ Some common mechanism that apply throughout the language.

Once you have grasped these ideas, you may be able to read. UML create some basic ones. As you gain more experience in applying conceptual model using more advanced features of this language.

## **Building blocks of the UML:**

The vocabulary of UML encompasses these kinds of building blocks.

### **❖ Use CASE definition:**

#### **Description:**

A use case is a set of scenarios tied together by a common user goal. A use case is a behavioral diagram that shows a set of use case actions and their relationships.

#### **Purpose:**

The purpose of use case is login and exchange messages between sender and receiver (Email client).

#### **Main flow:**

First, the sender gives his id and enters his login. Now, he enters the message to the receiver id.

#### **Alternate flow:**

If the username and id by the sender or receiver is not valid, the administrator will not allow entering and “Invalid password” message is displayed.

#### **Pre-condition:**

A person has to register himself to obtain a login ID.

#### **Post-condition:**

The user is not allowed to enter if the password or user name is not valid.

## ❖ **Class diagram:**

### **Description:**

☞ A class diagram describes the type of objects in system and various kinds of relationships that exists among them.

☞ Class diagrams and collaboration diagrams are alternate representations of object models.

During analysis, we use class diagram to show roles and responsibilities of entities that provide email client system behaviors design. We use to capture the structure of classes that form the email client system architecture.

### **A class diagram is represented as:**

<<Class name>>

<<Attribute 1>>

<<Attribute n>>

<<Operation ()>>

### **Relationship used:**

A change in one element affects the other.

### **Generalization:**

It is a kind of relationship.

### **State chart:**

#### **Description:**

☞ The state chart diagram made the dynamic behavior of individual classes.

☞ State chart shows the sequences of states that an object goes through events and state transitions.

☞ A state chart contains one state „start“ and multiple „end“ states.

The important objectives are:

**Decision:**

It represents a specific location state chart diagram where the work flow may branch based upon guard conditions.

**Synchronization:**

It gives a simultaneous workflow in a state chart diagram. They visually define forks and joins representing parallel workflow.

**Forks and joins:**

- ☐ A fork construct is used to model a single flow of control.
- ☐ Every work must be followed by a corresponding join.
- ☐ Joints have two or more flow that unit into a single flow.

**State:**

A state is a condition or situation during a life of an object in which it satisfies condition or waits for some events.

**Transition:**

It is a relationship between two activities and between states and activities.

**Start state:**

A start state shows the beginning of a workflow or beginning of a state machine on a state chart diagram.

**End state:**

It is a final or terminal state.

## ❖ **Activity diagram:**

### **Description:**

Activity diagram provides a way to model the workflow of a development process. We can also model this code specific information such as class operation using activity diagram. Activity diagrams can model different types of diagrams. There are various tools involved in the activity diagram.

### **Activity:**

An activity represents the performance of a task on duty. It may also represent the execution of a statement in a procedure.

### **Decision:**

A decision represents a condition on situation during the life of an object, which it satisfies some condition or waits for an event.

### **Start state:**

It represents the condition explicitly the beginning of a workflow on an activity.

### **Object flow:**

An object on an activity diagram represents the relationship between activity and object that creates or uses it.

### **Synchronization:**

It enables us to see a simultaneous workflow in an activity.

### **End state:**

An end state represents a final or terminal state on an activity diagram or state chart diagram.

## ❖ **Sequence diagram:**

### **Description:**

A sequence diagram is a graphical view of scenario that shows object interaction in a time based sequence what happens first what happens next. Sequence diagrams are closely related to collaboration diagram. The main difference between sequence and collaboration diagram is that sequence diagram show time based interaction while collaboration diagram shows objects associated with each other.

The sequence diagram for the e-mail client system consists of the following objectives:

### **Object:**

An object has state, behavior and identity. An object is not based is referred to as an instance. The various objects in e-mail client system are:

- ☐ User
- ☐ Website
- ☐ Login
- ☐ Groups

### **Message icon:**

A message icon represents the communication between objects indicating that an action will follow. The message icon is the horizontal solid arrow connecting lifelines together.

## **Collaboration diagram:**

### **Description:**

Collaboration diagram and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram shows objects, their links and their messages. They can also contain simple class instances and class utility instances.

During, analysis indicates the semantics of the primary and secondary interactions. Design, shows the semantics of mechanisms in the logical design of system.

Toggling between the sequence and collaboration diagrams.

When we work in either a sequence or collaboration diagram, it is possible to view the corresponding diagram by pressing F5 key.

## **CONFERENCE MANAGEMENT SYSTEM**

### **PROBLEMSTATEMENT:**

The process of the candidates is to login the conference system and submit the paper through online. Then the reviewer reviews the paper and sends the acknowledgement to the candidate either paper selected or rejected. This process of on conference management system are described sequentially through following steps,

- The candidate login to the conference management system.
- The paper title is submitted.
- The paper is been reviewed by the reviewer.
- The reviewer sends acknowledgement to the candidate.
- Based on the selection, the best candidate is selected.
- Finally the candidate registers all details.

### **SOFTWARE REQUIREMENT SPECIFICATION :**

#### **CANDIDATE**

The candidate can login and submit the paper to the reviewer. After getting acknowledgement the candidate will submit the revised and camera ready paper then registration process will be carried out.

#### **REVIEWER**

Reviewer will reviews the paper and sending acknowledgement to the candidate

#### **DATABASE**

Database is used to verify login and store the details of selected candidates.



## **SOFTWARE REQUIREMENT SPECIFICATION**

This software specification documents full set of features and function for conference management system.

### **PURPOSE**

The purpose of the conference management system is that the system can easily review the process. The main process in this document is the submission of paper by the candidate, reviewing process by the reviewer and sending of acknowledgement to the candidates whose paper is selected.

### **SCOPE**

The scope of this conference management process is to select the best candidate from the list of candidates based on their performance in the process.

### **FUNCTIONALITY**

The main functionality of conference system is to select the candidate for the presentation in conference.

### **USABILITY**

The user interface to make the process should be effective that is the system will help the candidates to register easily. The system should be user friendly.

### **PERFORMANCE**

It describes the capability of the system to perform the conference process of the candidate without any error and performing it efficiently.

## **RELIABILITY**

The conference system should be able to serve the applicant with correct information and day-to-day update of information.

## **FUNCTIONAL REQUIREMENTS**

Functional requirements are those that refer to the functionality of the system that is the services that are provided to the candidate who register for the conference.

## **UML DIAGRAMS :**

The following UML diagrams describe the process involved in the conference management system.

## **USE CASE DIAGRAM**

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It is represented using ellipse. Actor is any external entity that makes use of the system being modeled. It is represented using stick figure.

## **DOCUMENTATION OF USE CASE DIAGRAM**

The actors in this use case diagram are candidate, reviewer and database.

The use cases are the activities performed by actors.

The actors in this use case diagram are

- **Candidate** - Logins the conference system and submits the paper then do the registration process.

- **Reviewer** – Review the paper , select best candidate and send acknowledgement to them.
- **Databases** - verify the login and register details and selected candidate details are stored in it.

The use cases in this use case diagram are

- **Login** - Candidate enter their username and password to login to the conference system.

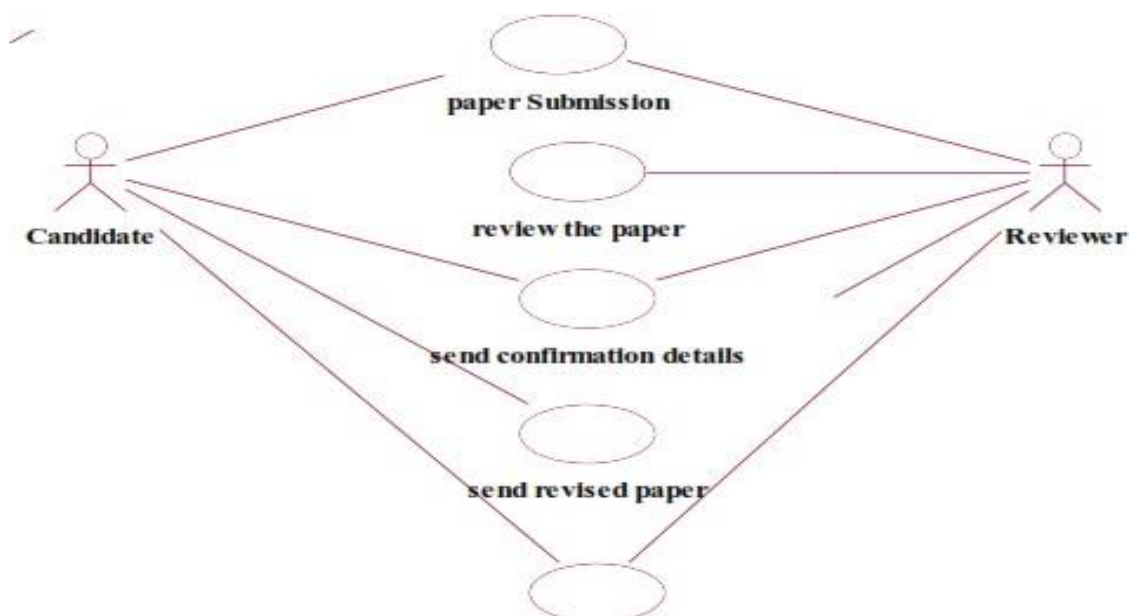
**Paper submission**– Candidate submits the paper.

- **Review the paper**– The paper is been reviewed by the reviewer and the paper is selected.

- **Paper confirmation details** – The reviewer can send the confirmation details to the candidate.

- **Revised and camera ready paper** – After the paper is selected and the camera ready paper should be submitted to the reviewer by candidate.

- **Registration** – After submitting the revised paper the candidate wants to register.



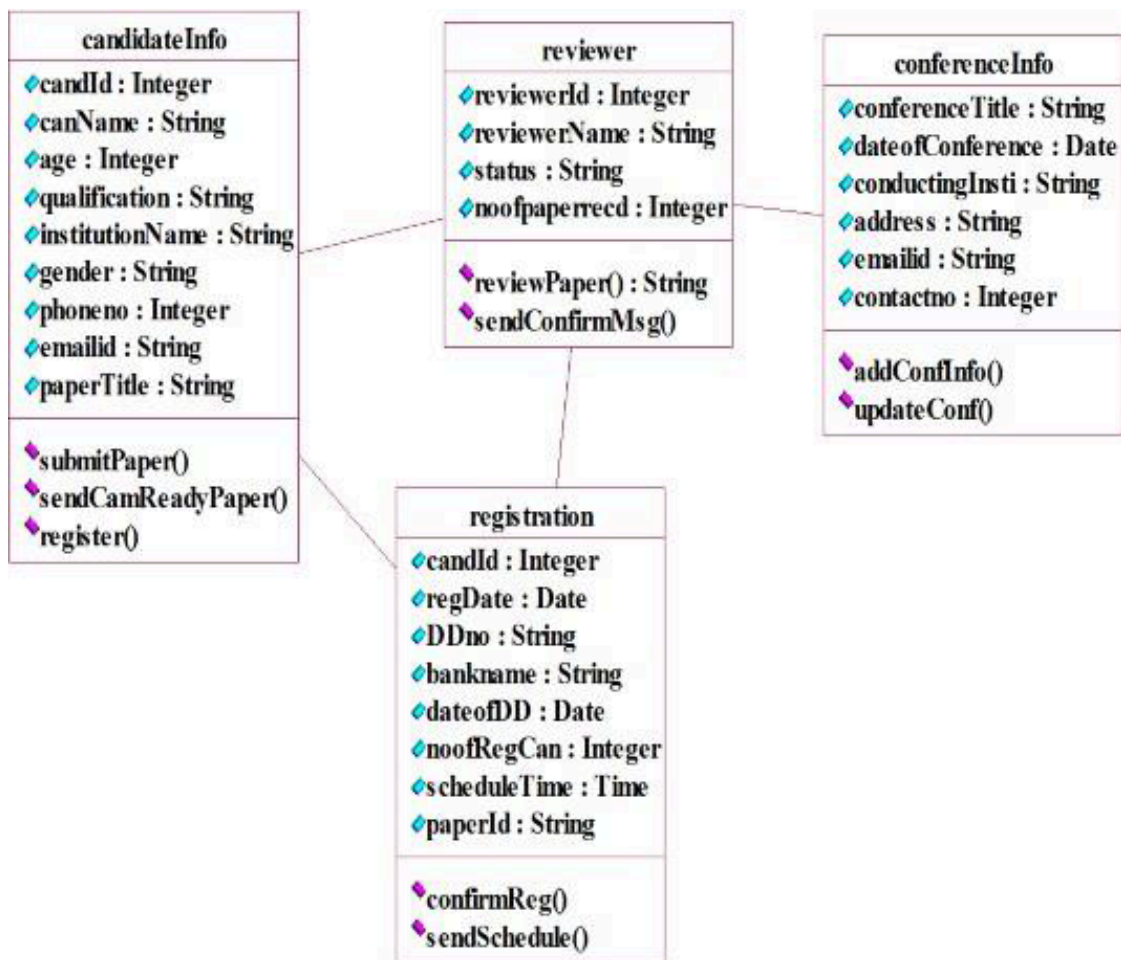
## **CLASS DIAGRAM**

A class diagram in the unified modeling language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. It is represented using a rectangle with three compartments. Top compartment have the class name, middle compartment the attributes and the bottom compartment with operations.

## **DOCUMENTATION OF CLASS DIAGRAM**

This class diagram has three classes candidate, reviewer and database.

- **Candidate** – Its attributes are name ,collegename , department , paper title. The operations performed in the candidate class are login, submit the paper, submit revised and camera ready paper and registration.
- **Reviewer** – Its attributes are name, department, reviewer ID The operations performed are review the paper and send the paper confirmation details.
- **Database** –The operations performed are storing candidate details and verifying login .

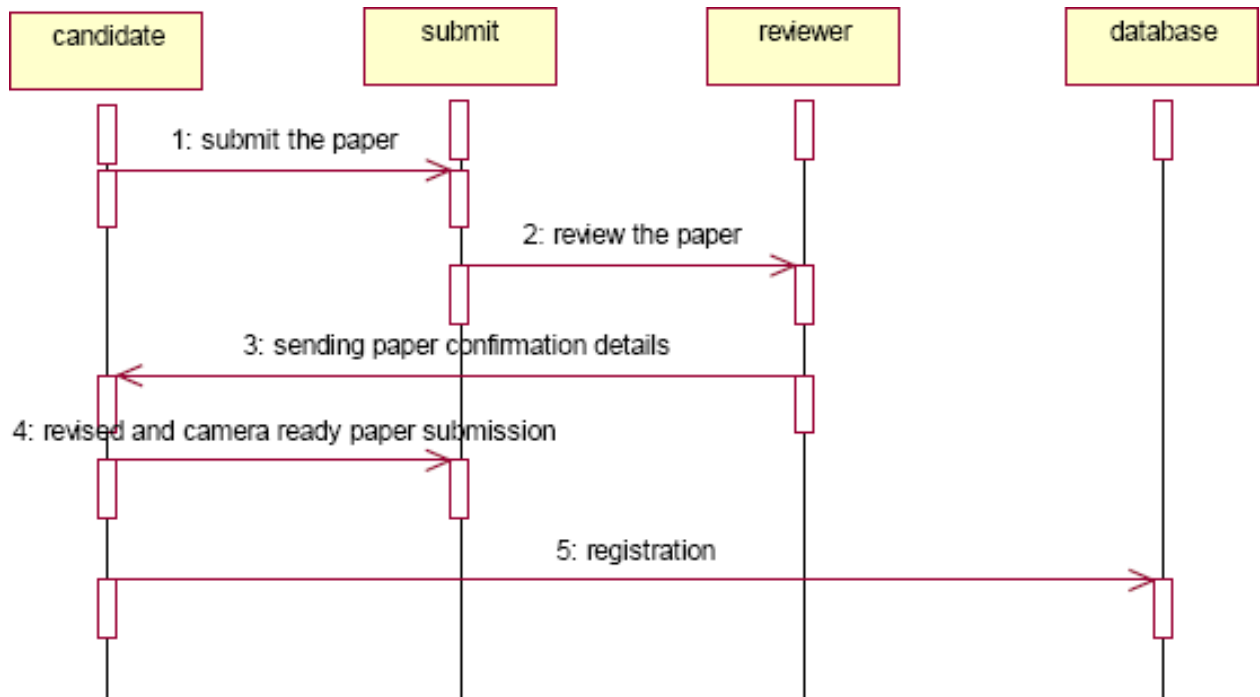


## SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. There are two dimensions.

1. Vertical dimension-represent time.
2. Horizontal dimension-represent different objects.

## SEQUENCE DIAGRAM FOR SUBMITTING PAPER



## DOCUMENTATION OF SEQUENCE DIAGRAM LOGIN

This sequence diagram describes the sequence of steps to show

- The candidate login in to the conference system and register for job.
- The verification done in the database .

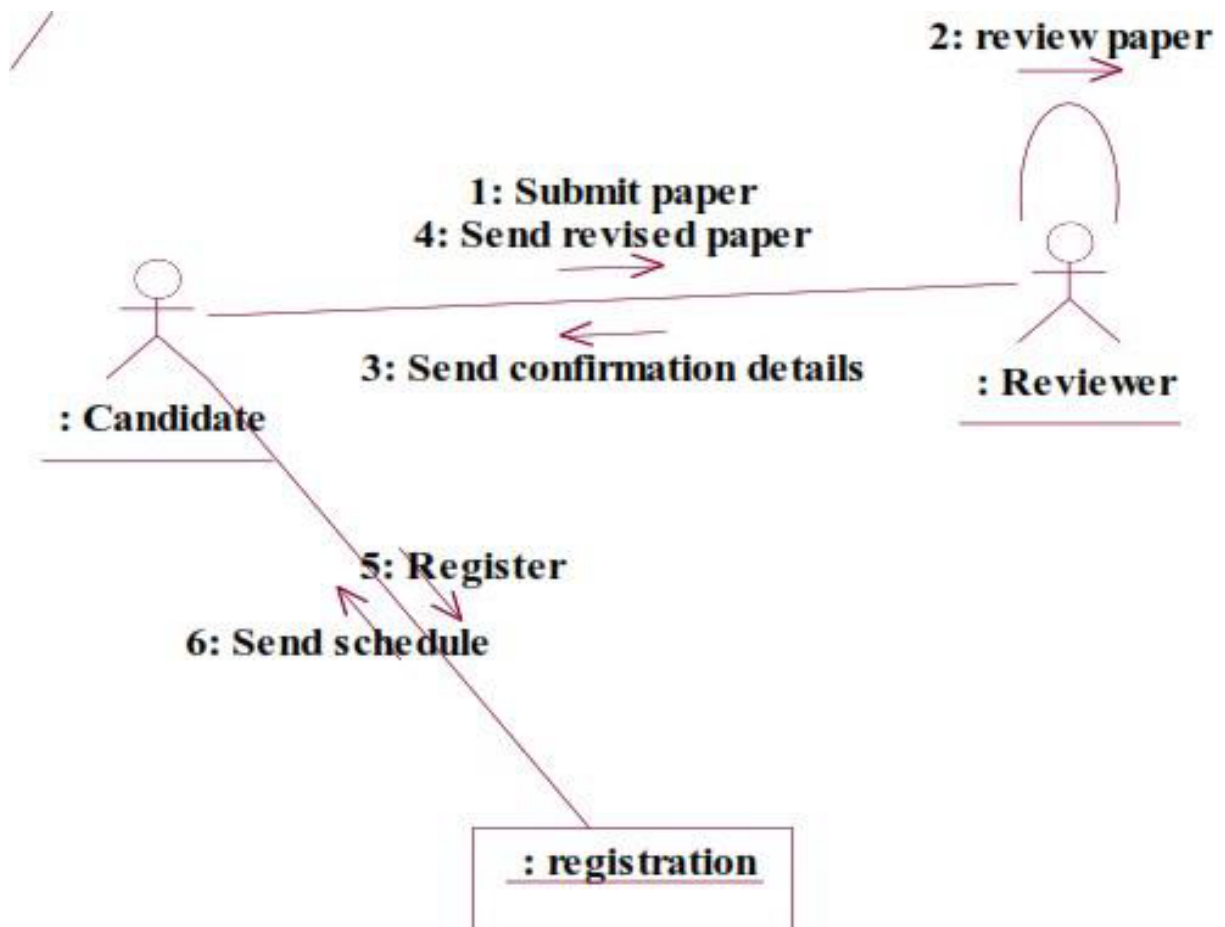
## PAPER SUBMISSION

This sequence diagram shows steps to show

- The candidate sumbit the paper.
- The reviewer reviews the paper and sends acknowledgement to the candidate.
- The candidate submits revised and camera ready paper.
- This candidate will registers their detials.

## COLLABRATION DIAGRAM

A collaboration diagram, also called a communication diagram or interaction diagram,. A sophisticated modeling tool can easily convert a collaboration diagram into a sequence diagram and the vice versa. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time.



## **DOCUMENTATION OF COLLABRATION DIAGRAM LOGIN**

This collaboration diagram is to show how the applicant login in the conference system. Here the sequence is numbered according to the flow of execution.

## **PAPER SUBMISSION**

This collaboration diagram is to show the submitting paper process of the candidate for the conference. The flow of execution of this selection process is represented using the numbers.

## **STATE CHART DIAGRAM**

The purpose of state chart diagram is to understand the algorithm involved in performing a method. It is also called as state diagram. A state is represented as a round box, which may contain one or more compartments. An initial state is represented as small dot. A final state is represented as circle surrounding a small dot.

## **DOCUMENTATION OF STATE CHART DIAGRAM**

This state diagram describes the behaviour of the system.

- First state is login where the candidate login to the conference system.
- The next state is submitting the paper .
- Then review the paper if it is selected the process will continue..
- The candidate should submit revised and camera ready paper.



## **ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. An activity is shown as an rounded box containing the name of the operation.

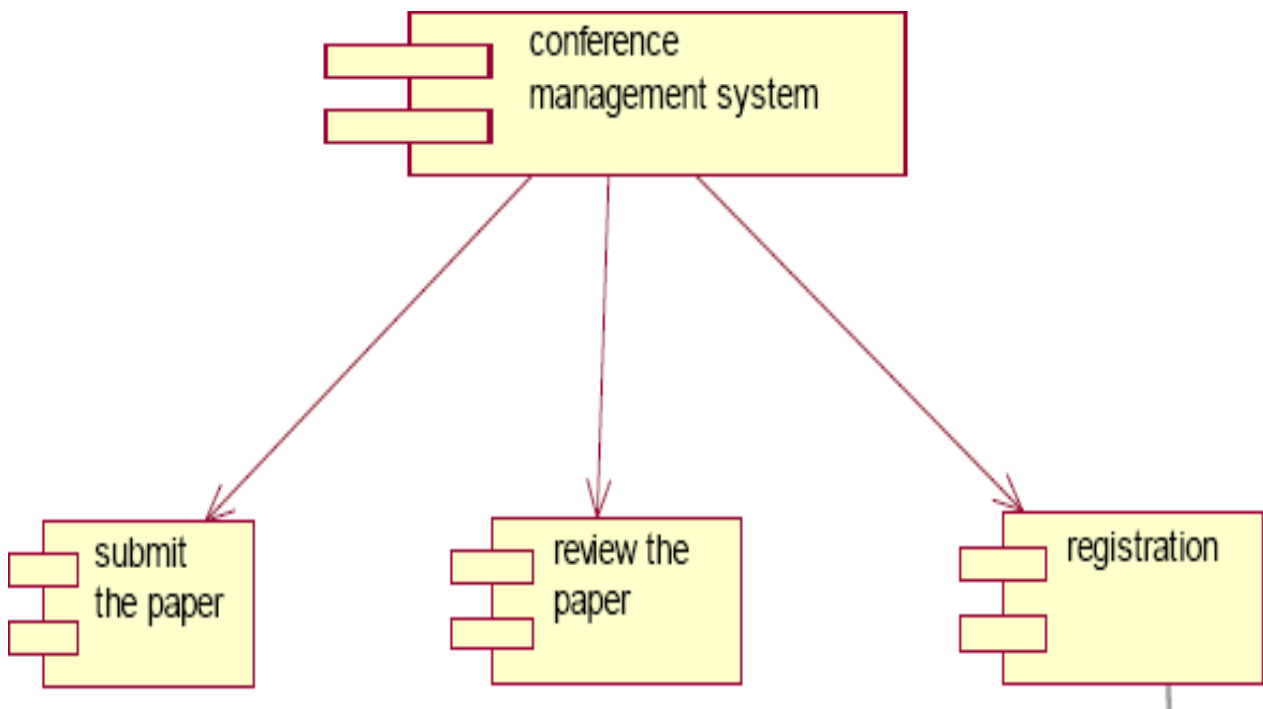
## **DOCUMENTATION OF ACTIVITY DIAGRAM**

This activity diagram flow of stepwise activities performed in recruitment system.

- First the candidate login to the database.
- Then the candidate should submit the paper.
- If it is selected the acknowledgement will send to the candidate.
- After submitting revised paper the registration proces will be done.

## COMPONENT DIAGRAM

The component diagram's main purpose is to show the structural relationships between the components of a system. It is represented by boxed figure. Dependencies are represented by communication association.

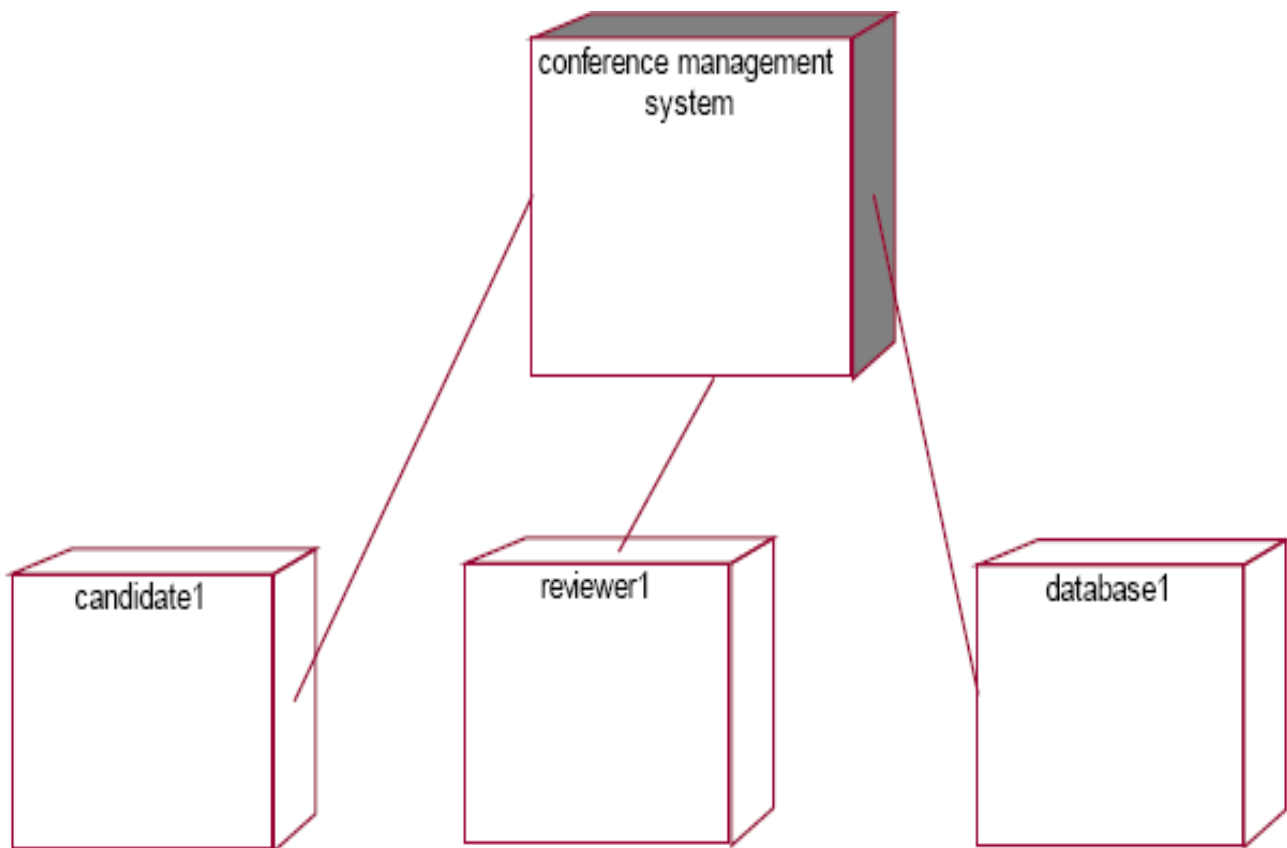


## DOCUMENTATION OF COMPONENT DIAGRAM

The main component in this component diagram is conference management system. And submit the paper, review the paper and registration.

## DEPLOYMENT DIAGRAM

A deployment diagram in the unified modeling language serves to model the physical deployment of artifacts on deployment targets. Deployment diagrams show "the allocation of artifacts to nodes according to the Deployments defined between them. It is represented by 3-dimensional box. Dependencies are represented by communication association.



## **DOCUMENTATION OF DEPLOYMENT DIAGRAM**

The processor in this deployment diagram is the conference management system which is the main part and the devices are the candidate, appear for do conference , reviewer will reviews paper , database will store all details which are the some of the main activities performed in the system.

## **PACKAGE DIAGRAM**

A package diagram in unified modeling language that depicts the dependencies between the packages that make up a model. A Package Diagram (PD) shows a grouping of elements in the OO model, and is a Cradle extension to UML. PDs can be used to show groups of classes in Class Diagrams (CDs), groups of components or processes in Component Diagrams (CPDs), or groups of processors in Deployment Diagrams (DPDs).

## **DOCUMENTATION OF PACKAGE DIAGRAM**

The three layers in the online recruitment system are

- **The User interface layer** - consists of the web and login. This layer describes how the candidate login.
- **The Domain layer** – shows the activities that are performed in the conference management system. The activities are paper submission , review paper , registration.
- **The Technical service layer** - the verification details and the selected candidate details will stored into the database.

## **PROGRAM CODE:**

```
# conference_management_system.py
```

```
from flask import Flask, render_template, request, redirect, url_for,  
flash, session, send_from_directory
```

```
from werkzeug.utils import secure_filename
```

```
from werkzeug.security import generate_password_hash,  
check_password_hash
```

```
import os
```

```
import sqlite3
```

```
app = Flask(__name__)
```

```
app.secret_key = 'your_secret_key'
```

```
app.config['UPLOAD_FOLDER'] = 'uploads'
```

```
if not os.path.exists('uploads'):
```

```
    os.makedirs('uploads')
```

```
# Database initialization
```

```
def init_db():
```

```
    with sqlite3.connect('conference.db') as conn:
```

```
        cursor = conn.cursor()
```

```
        cursor.execute("""
```

```
            CREATE TABLE IF NOT EXISTS users (
```

```
                id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
                name TEXT NOT NULL,
```

```
                email TEXT UNIQUE NOT NULL,
```

```
                password TEXT NOT NULL,
```

```
                role TEXT NOT NULL
```

```
            )
```

```

    '')
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS papers (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT NOT NULL,
            abstract TEXT,
            filename TEXT,
            status TEXT DEFAULT 'Pending',
            candidate_id INTEGER,
            FOREIGN KEY(candidate_id) REFERENCES users(id)
        )
    ''')
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS reviews (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            paper_id INTEGER,
            reviewer_id INTEGER,
            decision TEXT,
            comments TEXT,
            FOREIGN KEY(paper_id) REFERENCES papers(id),
            FOREIGN KEY(reviewer_id) REFERENCES users(id)
        )
    ''')

```

**init\_db()**

**# Routes and views**

**@app.route('/')**

**def index():**

```
    return render_template('index.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = generate_password_hash(request.form['password'])
        role = request.form['role']

        with sqlite3.connect('conference.db') as conn:
            cursor = conn.cursor()
            try:
                cursor.execute("INSERT INTO users (name, email, password,
role) VALUES (?, ?, ?, ?)", (name, email, password, role))
                conn.commit()
                flash('Registration successful. Please log in.', 'success')
                return redirect(url_for('login'))
            except sqlite3.IntegrityError:
                flash('Email already registered.', 'danger')

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
```

```
with sqlite3.connect('conference.db') as conn:
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users WHERE email = ?",
(email,))
    user = cursor.fetchone()
    if user and check_password_hash(user[3], password):
        session['user_id'] = user[0]
        session['name'] = user[1]
        session['email'] = user[2]
        session['role'] = user[4]
        return redirect(url_for('dashboard'))
    else:
        flash('Invalid credentials', 'danger')
```

```
return render_template('login.html')
```

```
@app.route('/logout')
```

```
def logout():
```

```
    session.clear()
```

```
    flash('Logged out successfully.', 'success')
```

```
    return redirect(url_for('index'))
```

```
@app.route('/dashboard')
```

```
def dashboard():
```

```
    if 'user_id' not in session:
```

```
        return redirect(url_for('login'))
```

```
    if session['role'] == 'candidate':
```

```
        with sqlite3.connect('conference.db') as conn:
```



```
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM papers WHERE
candidate_id = ?", (session['user_id'],))
        papers = cursor.fetchall()
        return render_template('candidate_dashboard.html',
papers=papers)
```

```
elif session['role'] == 'reviewer':
    with sqlite3.connect('conference.db') as conn:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM papers WHERE status =
'Pending'")
        papers = cursor.fetchall()
        return render_template('reviewer_dashboard.html',
papers=papers)
```

```
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        title = request.form['title']
        abstract = request.form['abstract']
        file = request.files['file']
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

        with sqlite3.connect('conference.db') as conn:
            cursor = conn.cursor()
            cursor.execute("INSERT INTO papers (title, abstract, filename,
candidate_id) VALUES (?, ?, ?, ?)",
                (title, abstract, filename, session['user_id']))
```

```

        conn.commit()

        flash('Paper submitted successfully.', 'success')
        return redirect(url_for('dashboard'))

    return render_template('submit.html')

@app.route('/review/<int:paper_id>', methods=['GET', 'POST'])
def review(paper_id):
    if session['role'] != 'reviewer':
        flash('Access denied.', 'danger')
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        decision = request.form['decision']
        comments = request.form['comments']

        with sqlite3.connect('conference.db') as conn:
            cursor = conn.cursor()

            cursor.execute("INSERT INTO reviews (paper_id, reviewer_id,
decision, comments) VALUES (?, ?, ?, ?)",
                (paper_id, session['user_id'], decision, comments))

            cursor.execute("UPDATE papers SET status = ? WHERE id =
?", (decision, paper_id))

            conn.commit()

            flash('Review submitted.', 'success')
            return redirect(url_for('dashboard'))

        with sqlite3.connect('conference.db') as conn:
            cursor = conn.cursor()

            cursor.execute("SELECT * FROM papers WHERE id = ?",

```

```
(paper_id,))  
    paper = cursor.fetchone()  
    return render_template('review.html', paper=paper)  
  
@app.route('/uploads/<filename>')  
def uploaded_file(filename):  
    return send_from_directory(app.config['UPLOAD_FOLDER'],  
filename)  
  
if __name__ == '__main__':  
    app.run(debug=True)
```

### **INDEX.html:**

```
<!DOCTYPE html>  
  
<html>  
  
<head><title>Conference Management</title></head>  
  
<body>  
    <h1>Welcome to the Conference System</h1>  
    <a href="/login">Login</a> | <a href="/register">Register</a>  
  
</body>  
  
</html>
```

## REGISTER.html :

<!DOCTYPE html>

<html>

<head><title>Register</title></head>

<body>

<h2>Register</h2>

<form method="POST">

Name: <input name="name" required><br>

Email: <input name="email" type="email" required><br>

Password: <input name="password" type="password" required><br>

Role:

<select name="role">

<option value="candidate">Candidate</option>

<option value="reviewer">Reviewer</option>

</select><br>

<button type="submit">Register</button>

</form>

<a href="/login">Already have an account? Login</a>

</body>

</html>

## LOGIN.html :

```
<!DOCTYPE html>

<html>

<head><title>Login</title></head>

<body>

    <h2>Login</h2>

    <form method="POST">

        Email: <input name="email" type="email" required><br>

        Password: <input name="password" type="password"
required><br>

        <button type="submit">Login</button>

    </form>

    <a href="/register">Don't have an account? Register</a>

</body>

</html>
```

## CANDITATE\_DASHBOARD.html

```
<!DOCTYPE html>

<html>

<head><title>Candidate Dashboard</title></head>

<body>

    <h2>Welcome {{ session.name }}</h2>

    <a href="/submit">Submit New Paper</a> | <a
href="/logout">Logout</a>

    <h3>Your Submissions</h3>
```

```

<ul>

    {% for paper in papers %}

        <li><b>{{ paper[1] }}</b> - Status: {{ paper[4] }}</li>

    {% endfor %}

</ul>

</body>

</html>

```

## REVIEWER\_DASHBOARD.html

```

<!DOCTYPE html>

<html>

<head><title>Reviewer Dashboard</title></head>

<body>

    <h2>Welcome {{ session.name }}</h2>

    <a href="/logout">Logout</a>

    <h3>Pending Papers</h3>

    <ul>

        {% for paper in papers %}

            <li>

                <b>{{ paper[1] }}</b> - <a href="{{ url_for('review',
paper_id=paper[0]) }}">Review</a>

            </li>

        {% endfor %}

    </ul>

</body>

```

</html>

## SUBMIT\_DASHBOARD.html

<!DOCTYPE html>

<html>

<head><title>Submit Paper</title></head>

<body>

<h2>Submit Paper</h2>

<form method="POST" enctype="multipart/form-data">

Title: <input name="title" required><br>

Abstract: <textarea name="abstract" required></textarea><br>

File: <input type="file" name="file" required><br>

<button type="submit">Submit</button>

</form>

</body>

</html>

## REVIEW.html

<!DOCTYPE html>

<html>

<head><title>Review Paper</title></head>

<body>

<h2>Review Paper: {{ paper[1] }}</h2>

<p><b>Abstract:</b> {{ paper[2] }}</p>

<a href="/uploads/{{ paper[3] }}" target="\_blank">Download  
PDF</a>

<form method="POST">

Decision:

<select name="decision">

<option value="Accepted">Accept</option>

<option value="Rejected">Reject</option>

</select><br>

Comments:<br>

<textarea name="comments" required></textarea><br>

<button type="submit">Submit Review</button>

</form>

</body>

</html>



## **OUTPUT:**

# **CANDIDATE REGISTER AND LOGIN AND SUBMITTING PAPER**


---

**Welcome to the Conference System**

[Login](#) | [Register](#)

---

### **Register**

Name:   
Email:   
Password:   
Role:    
  
[Already have an account? Login](#)

---

### Register

Name: Mohamed Aslam  
Email: xyz@gmail.com  
Password: \*\*\*\*\*  
Role: Candidate ▼  
  
[Already have an account? Login](#)

---

### Login

Email: xyz@gmail.com  
Password: \*\*\*\*\*  
  
[Don't have an account? Register](#)

### Welcome Aslam

[Submit New Paper](#) | [Logout](#)

### Your Submissions

---

## Submit Paper

Title:   

user interface, paper submissions, review outcomes, and registration information. The diagrams, including use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams, deployment diagrams, and package diagrams, are used to model various aspects of the system architecture and functionality. These diagrams provide a comprehensive understanding of the system's structure, interactions, and workflow. The Conference Management System project successfully implements an automated platform for managing conference submissions, reviews, and registrations, enhancing efficiency and convenience for all stakeholders involved.

Abstract:   
File:  Conference ...t system.odt

## Welcome Aslam

[Submit New Paper](#) | [Logout](#)

### Your Submissions

- conference\_management\_system - Status: Pending

---

# REVIEWER REGISTER AND LOGIN AND REVIEW THE SUBMITTED PAPER

## Register

Name: Imthiyaz

Email: pqr@gmail.com

Password: \*\*\*\*

Role: Reviewer

Register

Already have an account? Login

## Welcome Imthiyaz

Logout

### Pending Papers

- conference\_management\_system - Review

## Review Paper: conference\_management\_system

**Abstract:** The Conference Management System project aims to streamline the process of managing conference submissions, reviews, and registrations through an automated platform. This system facilitates the submission of papers by candidates, their review by designated reviewers, and the subsequent acknowledgment sent to the candidates. Key functionalities of the system include user authentication, paper submission, reviewer assignment, acknowledgment generation, candidate selection, and registration management. Candidates can easily submit their papers online, which are then reviewed by assigned reviewers. Based on the review, acknowledgments are sent to candidates, and selected candidates proceed with registration. The software interface is developed using Rational Rose Software, providing a user-friendly platform for candidates, reviewers, and administrators. The backend infrastructure utilizes a database to store user details, paper submissions, review outcomes, and registration information. UML diagrams, including use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams, deployment diagrams, and package diagrams, are used to model various aspects of the system architecture and functionality. These diagrams provide a comprehensive understanding of the system's structure, interactions, and workflow. The Conference Management System project successfully implements an automated platform for managing conference submissions, reviews, and registrations, enhancing efficiency and convenience for all stakeholders involved.

Download PDF

Decision: Accept

Comments:

Submit Review

---

### Review Paper: conference\_management\_system

**Abstract:** The Conference Management System project aims to streamline the process of managing conference submissions, reviews, and registrations through an automated platform. This system facilitates the submission of papers by candidates, their review by designated reviewers, and the subsequent acknowledgment sent to the candidates. Key functionalities of the system include user authentication, paper submission, reviewer assignment, acknowledgment generation, candidate selection, and registration management. Candidates can easily submit their papers online, which are then reviewed by assigned reviewers. Based on the review, acknowledgments are sent to candidates, and selected candidates proceed with registration. The software interface is developed using Rational Rose Software, providing a user-friendly platform for candidates, reviewers, and administrators. The backend infrastructure utilizes a database to store user details, paper submissions, review outcomes, and registration information. UML diagrams, including use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams, deployment diagrams, and package diagrams, are used to model various aspects of the system architecture and functionality. These diagrams provide a comprehensive understanding of the system's structure, interactions, and workflow. The Conference Management System project successfully implements an automated platform for managing conference submissions, reviews, and registrations, enhancing efficiency and convenience for all stakeholders involved.

[Download PDF](#)

Decision: Accept

Comments:

good one

[Submit Review](#)

Welcome Imthiyaz

[Logout](#)

Pending Papers

# CANDIDATE PAGE

**Welcome Aslam**

[Submit New Paper](#) | [Logout](#)

**Your Submissions**

- `conference_management_system` - Status: Accepted

## **CONCLUSION**

The Conference Management System provides a streamlined and efficient platform for managing the lifecycle of academic paper submissions, reviews, and results dissemination. Through a user-friendly web interface, candidates can register, submit research papers, and monitor the status of their submissions. Meanwhile, reviewers can access and evaluate submitted papers, making decisions that are reflected in real-time to the respective authors.

This system enhances transparency, reduces administrative overhead, and supports scalability for conferences of varying sizes. By leveraging Flask and SQLite, the project offers a lightweight yet functional prototype that can be extended further with features such as admin controls, automated email notifications, reviewer-paper assignment algorithms, and integration with citation databases.

In conclusion, this project demonstrates the practical application of web development skills to solve real-world academic coordination challenges and lays the foundation for a fully-featured academic event management platform