Problem Statement:

In the contemporary workspace, whether it be for individuals or collaborative teams, task management remains a critical and often cumbersome process. Traditional methods of task tracking through physical boards or disparate digital tools lack integration, leading to several pain points:

1. **Inefficient Task Tracking:** Many teams and individuals currently rely on manual or disjointed methods for tracking tasks, leading to inefficiencies and a lack of clear oversight.
2. **Lack of Centralized System:** Without a centralized platform, it's challenging to assign tasks, set deadlines, and monitor progress, resulting in decreased productivity and potential project delays.
3. **Difficulty in Collaboration:** Collaboration on tasks often requires multiple communication channels. The absence of a unified system complicates the sharing of updates and feedback.
4. **Inadequate Task Prioritization:** Prioritizing tasks based on urgency and importance is crucial. However, many existing systems don’t offer an effective way to manage task priorities.
5. **Limited Accessibility and Visibility:** Teams need real-time access to task statuses and updates. Limited visibility into task progression hinders effective decision-making.

Proposed System:

Our project aims to address these challenges by developing an all-encompassing task management system that:

1. **Streamlines task tracking and management**

   This means that the system will make it easy to create, assign, track, and manage tasks. It should provide a clear overview of all tasks, including their status, due dates, and priorities. It should also be easy to filter and sort tasks, and to generate reports.

2. **Provides a centralized platform for task assignment, updates, and collaboration**

This means that the system will provide a single place where all team members can go to view and manage their tasks, as well as to collaborate on tasks and projects. This can help to improve communication and coordination, and to reduce duplication of effort.

3. **Enhances team productivity and efficiency through intuitive design and user-friendly features**

   The system should be easy to use and navigate, even for users with limited technical experience.

4. **Facilitates effective task prioritization and project planning**

   The system should provide tools and features to help users prioritize their tasks and plan their projects effectively.

5. **Offers real-time visibility into task status and progress for all team members**

   The system should provide real-time updates on the status and progress of all tasks, so that everyone involved can stay informed and make necessary adjustments as needed. This can help to improve accountability and reduce the risk of missed deadlines.

Overall, the goal of the task management system is to create a more organized, efficient, and collaborative work environment for teams and individuals. By providing a centralized platform for task tracking, management, and collaboration, the system can help users to save time, reduce stress, and achieve their goals more effectively.

Hardware and Software Requirements:

Hardware Requirements

- **Operating System**: Any operating system can be used. However, we used Mac OS for the development of this project.

- Memory: 1GB RAM or higher is recommended for smooth operation.
- Processor: An Intel Core i5 processor or equivalent is recommended for efficient processing.

Software Requirements

- Frontend:
    - HTML5: Hypertext Markup Language, 5th version, is used for the structure and content of the web pages.
    - CSS3: Cascading Style Sheets, 3rd version, is used for the presentation of the web pages.
    - JavaScript: A scripting language that allows web pages to be interactive.
- Backend:
    - Flask: A Python web framework used for the server-side logic.
    - Flask-Bcrypt: A Flask extension that provides password hashing.
    - Flask-SQLAlchemy: A Flask extension that provides database management.
    - Flask-Login: A Flask extension that provides user authentication.
    - Python: A general-purpose programming language used for the backend logic.
- Database:
    - XAMPP: A free and open-source software package that includes Apache, MySQL, PHP, and Perl.
    - MySQL: A popular relational database management system.

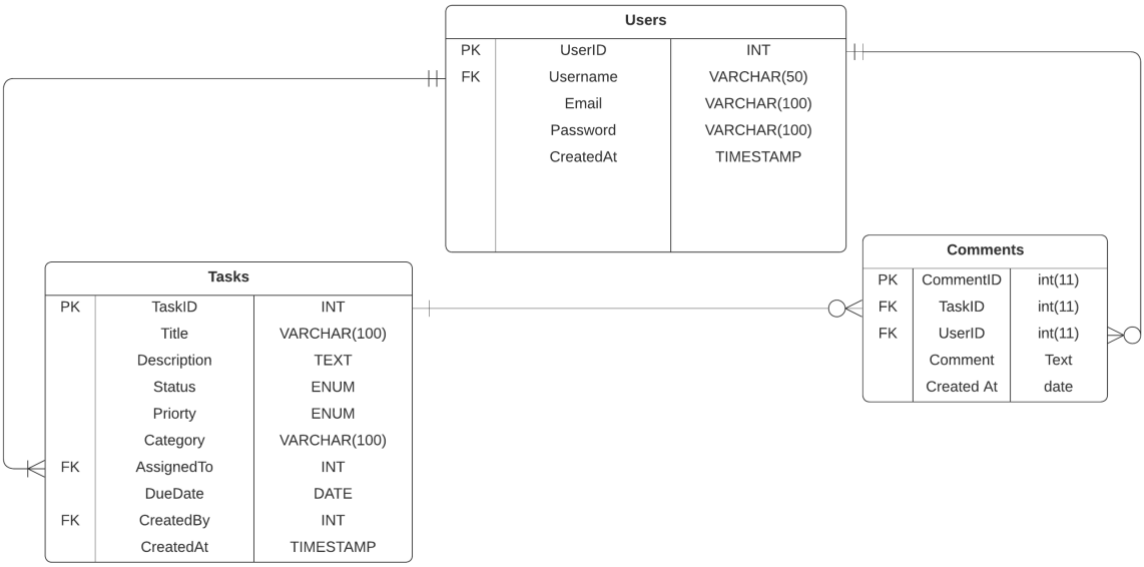
ER Diagram:

An entity-relationship diagram (ERD) is a graphical representation of the entities and their relationships in a database. It is used to model the data structure of a database and to communicate the design to stakeholders.

Entities - An entity is a person, place, thing, or event of interest in a database. Entities are represented in ERDs as rectangles. Each entity has a unique identifier, which is usually a primary key.

Relationships - A relationship is a connection between two entities. Relationships are represented in ERDs as diamonds. Each relationship has a name and a cardinality, which indicates the number of entities that can participate in the relationship on each side.

Below is the ER Diagram for the task management system.



ER Diagram for Task Management System

Schema Definition:

1. **Users Table**: Maintains user account details.
    - **UserID**: The primary key uniquely identifying each user.
    - **Username**: The user's chosen name, unique across the system.
    - **Email**: The user's email address, unique to ensure one account per email.
    - **Password**: A secure storage for the user's password.
    - **CreatedAt**: A timestamp marking the account creation time, defaults to the current timestamp.
2. **Tasks Table**: Holds information on individual tasks.

- **TaskID**: The primary key uniquely identifying each task.
- **Title**: The title of the task.
- **Description**: A text field describing the task.
- **Status**: An enumeration indicating whether the task is 'Pending', 'In Progress', or 'Completed'.
- **Priority**: An enumeration indicating the task's priority as 'Low', 'Medium', or 'High'.
- **Category**: A string indicating the task's category.
- **AssignedTo**: A foreign key linking to **UserID** indicating who is assigned to the task.
- **DueDate**: The date by which the task is due.
- **CreatedBy**: A foreign key linking to **UserID** indicating who created the task.
- **CreatedAt**: A timestamp for when the task was created, defaults to the current timestamp.

3. **Comments Table**: Stores comments made on tasks.
    - **CommentID**: The primary key uniquely identifying each comment.
    - **TaskID**: A foreign key that links to **TaskID** in the Tasks table, indicating the task to which the comment belongs.
    - **Comment**: A text field for the comment content.
    - **UserID**: A foreign key that links to **UserID** in the Users table, indicating who made the comment.
    - **CreatedAt**: A timestamp for when the comment was made, defaults to the current timestamp.

4. **Login Table**: An additional table that seems to replicate the Users table's functionality.
    - **id**: A unique identifier for each user's login, serving as the primary key.
    - **username**: The user's chosen username, unique to ensure no duplicates.
    - **email**: The user's email address, also unique.
    - **password_hash**: A hashed version of the user's password for security.

Relationships:

- The **Users** table is central to this schema, as it connects to both the **Tasks** and **Comments** tables. This establishes a one-to-many relationship from a single user to multiple tasks (both assigned and created) and comments.
- The **Tasks** table references the **Users** table twice: once for the user to whom the task is assigned (**AssignedTo**) and once for the user who created the task (**CreatedBy**). This reflects the workflow within a project management context.
- The **Comments** table connects comments to their respective tasks and the users who made them. This facilitates communication and collaboration on tasks within the system.
- The **Login** table's presence alongside the **Users** table suggests a possible architectural decision to separate authentication from user details. Each **Login** record would correspond to a **Users** record, but it is not clear how they are related without a foreign key reference.

These relationships enable the creation of a system that can track tasks, assign them to users, and facilitate discussion through comments, all while managing user accounts and authentication separately. This schema would support a project management application where tasks can be created, assigned, tracked, and discussed among team members.

MYSQL Code:

```sql
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: localhost
-- Generation Time: Dec 10, 2023 at 11:54 PM
-- Server version: 10.4.28-MariaDB
-- PHP Version: 8.2.4

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!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 */;

--
-- Database: `task_tracker`
--

-- --------------------------------------------------------

--
-- Table structure for table `Comments`
--

CREATE TABLE `Comments` (
  `CommentID` int(11) NOT NULL,
  `TaskID` int(11) NOT NULL,
  `Comment` text DEFAULT NULL,
  `UserID` int(11) DEFAULT NULL,
  `CreatedAt` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `Comments`
--

INSERT INTO `Comments` (`CommentID`, `TaskID`, `Comment`, `UserID`, `CreatedAt`) VALUES
(5, 2, 'This is a sample comment\r\n', 2, '2023-12-10 22:47:50'),
(6, 2, 'Xampp installation is pending', 2, '2023-12-10 22:48:47');

-- --------------------------------------------------------

--
-- Table structure for table `login`
--

CREATE TABLE `login` (
  `id` int(11) NOT NULL,
  `username` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password_hash` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `login`
--

INSERT INTO `login` (`id`, `username`, `email`, `password_hash`) VALUES
(1, 'Vignesh', 'garrapallyvignesh8055@gmail.com', '$2b$12$iMuE7uW/gUrfFhTlFmQK8OVLHdbfqoTQXX9S/ubUyDhFqIx3DAsAG'),
(2, 'Shirisha', 'Shirisha@gmail.com', '$2b$12$dhi/e3CIwYg8Ubby9pgUMeHHOpZ4TwVHEXdKzG4diWEyMofzq3uEK');

-- --------------------------------------------------------

--
-- Table structure for table `Tasks`
--

CREATE TABLE `Tasks` (
  `TaskID` int(11) NOT NULL,
  `Title` varchar(100) NOT NULL,
  `Description` text DEFAULT NULL,
  `Status` enum('Pending','In Progress','Completed') NOT NULL DEFAULT 'Pending',
  `Priority` enum('Low','Medium','High') NOT NULL DEFAULT 'Medium',
  `Category` varchar(100) DEFAULT NULL,
  `AssignedTo` int(11) DEFAULT NULL,
  `DueDate` date DEFAULT NULL,
  `CreatedBy` int(11) DEFAULT NULL,
  `CreatedAt` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `Tasks`
--

INSERT INTO `Tasks` (`TaskID`, `Title`, `Description`, `Status`, `Priority`, `Category`, `AssignedTo`, `DueDate`, `CreatedBy`, `CreatedAt`) VALUES
(2, 'Create Front end for database project', 'Add html code\r\ninstall ampp', 'Pending', 'High', 'Frontend', 2, '2023-12-12', 2, '2023-12-10 13:05:35'),
(3, 'Test task', 'Description', 'In Progress', 'Low', 'Database Systems', 2, '2023-12-13', 2, '2023-12-10 14:40:54'),
(4, 'Test task', 'Description', 'Completed', 'Low', 'Database Systems', 2, '2023-12-13', 2, '2023-12-10 14:41:00'),
(5, 'Test task', 'Description', 'Completed', 'Low', 'Database Systems', 2, '2023-12-13', 2, '2023-12-10 14:41:08');

-- --------------------------------------------------------
```

```sql
--
-- Table structure for table `Users`
--

CREATE TABLE `Users` (
  `UserID` int(11) NOT NULL,
  `Username` varchar(50) NOT NULL,
  `Email` varchar(100) NOT NULL,
  `Password` varchar(100) NOT NULL,
  `CreatedAt` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `Users`
--

INSERT INTO `Users` (`UserID`, `Username`, `Email`, `Password`, `CreatedAt`) VALUES
(2, 'abc', 'abc@gmail.com', '$2b$12$dtm2b9qa3gCHxZobph7ZnOHDkZa2XLuc0EaGxHn78FTRUO9k1t/FO', '2023-12-10 12:52:10'),
(3, 'Varun', 'varun@gamil.com', '$2b$12$3CLZOJAagGjhg5ytk5s2t.L.lkkMGinQdY/hk5OjNPELcakoB4hzy', '2023-12-10 17:19:47');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `Comments`
--
ALTER TABLE `Comments`
  ADD PRIMARY KEY (`CommentID`),
  ADD KEY `TaskID` (`TaskID`),
  ADD KEY `UserID` (`UserID`);

--
-- Indexes for table `login`
--
ALTER TABLE `login`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `username` (`username`),
  ADD UNIQUE KEY `email` (`email`);

--
-- Indexes for table `Tasks`
--
ALTER TABLE `Tasks`
  ADD PRIMARY KEY (`TaskID`),
  ADD KEY `AssignedTo` (`AssignedTo`),
  ADD KEY `CreatedBy` (`CreatedBy`);

--
-- Indexes for table `Users`
--
ALTER TABLE `Users`
  ADD PRIMARY KEY (`UserID`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `Comments`
--
ALTER TABLE `Comments`
  MODIFY `CommentID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;

--
-- AUTO_INCREMENT for table `login`
--
ALTER TABLE `login`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `Tasks`
--
ALTER TABLE `Tasks`
  MODIFY `TaskID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;

--
-- AUTO_INCREMENT for table `Users`
--
ALTER TABLE `Users`
  MODIFY `UserID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `Comments`
--
ALTER TABLE `Comments`
  ADD CONSTRAINT `comments_ibfk_1` FOREIGN KEY (`TaskID`) REFERENCES `Tasks` (`TaskID`),
  ADD CONSTRAINT `comments_ibfk_2` FOREIGN KEY (`UserID`) REFERENCES `Users` (`UserID`);

--
-- Constraints for table `Tasks`
--
ALTER TABLE `Tasks`
  ADD CONSTRAINT `tasks_ibfk_1` FOREIGN KEY (`AssignedTo`) REFERENCES `Users` (`UserID`),
  ADD CONSTRAINT `tasks_ibfk_2` FOREIGN KEY (`CreatedBy`) REFERENCES `Users` (`UserID`);
COMMIT;

/*!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 */;
```

Results:

Our task management system, TASK-TRACKER, has been meticulously designed to streamline the workflow of teams and individuals alike. The system provides a comprehensive solution for task management through a series of interconnected features, each with its own dedicated user interface.

1. **Admin Registration:** New Admins begin their journey with a straightforward sign-up process, ensuring secure access to the system with an intuitive interface.

   [image of signup]

2. **Secure Login:** Users can access their personalized dashboard through a secure login portal, safeguarding their information and workflow.

   [image of login]

3. **User Management:** Administrators have the capability to create new user accounts, enabling team expansion and role assignment with ease.

   [image of users page]

4. **Task Creation:** Users can create tasks with detailed descriptions, assign priorities, set due dates, and allocate them to team members. The task creation interface is designed for clarity and ease of use.

   [image of create task]

5. **Dashboard Overview:** Each person logged in is greeted by a dashboard that provides a comprehensive view of tasks sorted by status, allowing for quick assessments of individual and team progress.

   [image]

6. **Task Updating:** Tasks can be edited and updated to reflect changes in scope, assignment, or deadlines, ensuring the system remains current with the project's evolution**.**

   (Image Placeholder for Edit Task Page)

7. **Commenting Feature:** The system fosters collaboration by allowing users to add comments to tasks, facilitating communication and the sharing of ideas directly within the task's context.

   (Image Placeholder for Task Details with Comments)

Through this project, we have established a seamless workflow that enables users to manage tasks effectively from inception to completion.

Conclusion:

The TASK-TRACKER system has achieved its aim to provide an integrated environment for task management. It has successfully combined functionality with user-friendliness, creating a tool that not only improves productivity but also enhances the collaborative experience. Key outcomes include:

- **Increased Efficiency:** By centralizing task management, we have reduced the need for disparate tools and streamlined the task-tracking process.
- **Improved Visibility:** With real-time updates and a dashboard view, users and managers alike maintain a clear understanding of task statuses and project health.
- **Enhanced Collaboration:** The comment system and task update features have provided a platform for team members to communicate effectively and in context.
- **Secure and Scalable:** Security measures like hashed passwords and Flask's robust backend ensure the system is both secure and ready to scale with organizational growth.

In summary, TASK-TRACKER stands as a testament to the potential of well-designed software to transform the project management landscape. It offers a practical solution to common project management challenges, and its deployment marks a step forward in the pursuit of operational excellence.

Reference:

1. Apache Friends (XAMPP): Apache Friends. (n.d.). XAMPP. Retrieved December 11, 2023, from https://www.apachefriends.org/index.html

2. Flask-Login Documentation: Flask-Login. (n.d.). Flask-Login Documentation. Retrieved December 11, 2023, from https://flask-login.readthedocs.io/en/latest/

3. Flask-SQLAlchemy Documentation: Pallets Projects. (n.d.). Flask-SQLAlchemy Documentation. Retrieved December 11, 2023, from https://flask-sqlalchemy.palletsprojects.com/en/3.1.x/

4. Flask-Bcrypt Documentation: Flask-Bcrypt. (n.d.). Flask-Bcrypt Documentation. Retrieved December 11, 2023, from https://flask-bcrypt.readthedocs.io/en/1.0.1/

APPENDIX:

/app/routes.py

```
from flask import render_template, request, redirect, url_for, flash
from app import app, bcrypt
from app.models import  db, Login, User, Task, Comment
from flask_login import login_user, logout_user, login_required, current_user
from sqlalchemy.exc import SQLAlchemyError
from datetime import datetime

@app.route('/')
def home():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))
    return render_template('index.html')


@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = Login.query.filter_by(username=username).first()

        if user and user.check_password(password):
            login_user(user)
            return redirect(url_for('dashboard'))
        else:
            flash('Invalid username or password')
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))
    return render_template('login.html')

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

```
        existing_user = Login.query.filter_by(username=username).first()
        existing_email = Login.query.filter_by(email=email).first()
        if existing_user is None and existing_email is None:
            new_user = Login(username=username, email=email)
            new_user.set_password(password)
            db.session.add(new_user)
            db.session.commit()
            return redirect(url_for('login'))
        elif existing_user is not None:
            flash('Username already exists')
        elif existing_email is not None:
            flash('Email already exists')
        else:
            flash('Something went wrong')
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))
    return render_template('signup.html')

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('home'))

@app.route('/dashboard', methods=['GET', 'POST'])
@login_required
def dashboard():
    user_filter = request.args.get('user', 'all')
    priority_filter = request.args.get('priority', 'all')

    query = Task.query.join(User, Task.assigned_to == User.id).add_columns(
        Task.id, Task.title, Task.description, Task.status, Task.priority, Task.category, Task.assigned_to,
Task.due_date,
        User.username.label('assigned_username')
    )

    if user_filter != 'all':
        query = query.filter(Task.assigned_to == user_filter)
    if priority_filter != 'all':
```

```
        query = query.filter(Task.priority == priority_filter)

    tasks = query.all()
    users = User.query.all()
    #get the username for user_filter if it is not all, else set it to all
    if user_filter != 'all':
        user = User.query.get_or_404(user_filter).username
    else:
        user = 'all'
    return render_template('dashboard.html', tasks=tasks, users=users,user_filter=user, priority_filter=priority_filter)


@app.route('/users', methods=['GET', 'POST'])
@login_required
def users():
    users = User.query.all()
    return render_template('users.html', users=users)

@app.route('/create_user', methods=['GET', 'POST'])
@login_required
def create_user():
    if request.method == 'POST':
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')

        # Check if user already exists
        existing_user = User.query.filter_by(email=email).first()
        if existing_user:
            flash('Email already in use.', 'danger')
            return redirect(url_for('create_user'))

        # Create new user
        new_user = User(username=username, email=email,
password=bcrypt.generate_password_hash(password).decode('utf-8'))
        db.session.add(new_user)
        db.session.commit()

        flash('User created successfully!', 'success')
        return redirect(url_for('users'))
```

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


@app.route('/delete_user/<int:user_id>', methods=['POST'])
@login_required
def delete_user(user_id):
    try:
        user = User.query.get_or_404(user_id)
        db.session.delete(user)
        db.session.commit()
        flash('User deleted successfully!', 'success')
    except SQLAlchemyError as e:
        flash('Error deleting user!', 'danger')
    return redirect(url_for('users'))


@app.route('/create_task', methods=['GET', 'POST'])
@login_required
def create_task():
    users = User.query.all()

    if request.method == 'POST':
        title = request.form.get('title')
        description = request.form.get('description')
        status = request.form.get('status')
        priority = request.form.get('priority')
        category = request.form.get('category')
        assigned_to = request.form.get('assignedTo')
        due_date = request.form.get('dueDate')
        created_by = request.form.get('createdBy')
        new_task = Task(
            title=title, description=description, status=status, priority=priority,
            category=category, assigned_to=assigned_to, created_by=created_by,
            due_date=due_date
        )
        db.session.add(new_task)
        db.session.commit()

        flash('Task created successfully!', 'success')
```

```
        return redirect(url_for('dashboard'))

    return render_template('create_task.html', users=users)


@app.route('/task_details/<int:task_id>')
@login_required
def task_details(task_id):
    query = Task.query.join(User, Task.assigned_to == User.id).add_columns(
        Task.id, Task.title, Task.description, Task.status, Task.priority, Task.category, Task.assigned_to,
Task.due_date,
        User.username.label('assigned_username')
    )
    task = query.filter(Task.id == task_id).first()
    comments = Comment.query.filter_by(task_id=task_id).all()
    users = User.query.all()
    return render_template('task_details.html', task=task, comments=comments,users=users)

@app.route('/edit_task/<int:task_id>', methods=['GET', 'POST'])
@login_required
def edit_task(task_id):
    task = Task.query.get_or_404(task_id)
    users = User.query.all()
    if request.method == 'POST':
        try:
            task.title = request.form.get('title')
            task.description = request.form.get('description')
            task.priority = request.form.get('priority')
            task.category = request.form.get('category')
            task.assigned_to = request.form.get('assignedTo')
            db.session.commit()
            flash('Task updated successfully!', 'success')
        except SQLAlchemyError as e:
            flash('Error updating task!', 'danger')
        return redirect(url_for('task_details', task_id=task_id))
    return render_template('edit_task.html', task=task, users=users)


@app.route('/add_comment/<int:task_id>', methods=['POST'])
@login_required
```

```
def add_comment(task_id):
    comment_text = request.form.get('comment')
    user_id = request.form.get('user_id')
    new_comment = Comment(task_id=task_id, comment_text=comment_text, user_id=user_id,
created_at=datetime.now())
    db.session.add(new_comment)
    db.session.commit()
    return redirect(url_for('task_details', task_id=task_id))


@app.route('/delete_comment/<int:comment_id>', methods=['POST', 'GET'])
@login_required
def delete_comment(comment_id):
    try:
        comment = Comment.query.get_or_404(comment_id)
        db.session.delete(comment)
        db.session.commit()
        flash('Comment deleted successfully!', 'success')
    except SQLAlchemyError as e:
        flash('Error deleting comment!', 'danger')
    return redirect(url_for('dashboard'))


@app.route('/delete_task/<int:task_id>', methods=['POST', 'GET'])
@login_required
def delete_task(task_id):
    try:
        task = Task.query.get_or_404(task_id)
        print(task)
        db.session.delete(task)
        db.session.commit()
        flash('Task deleted successfully!', 'success')
    except SQLAlchemyError as e:
        flash('Error deleting task! Delete Comments First', 'danger')
    return redirect(url_for('dashboard'))
```

/app/models.py

```python
from datetime import datetime
from app import db, app, bcrypt
from flask_login import UserMixin

class User(db.Model):
    __tablename__ = 'Users'

    id = db.Column('UserID', db.Integer, primary_key=True)
    username = db.Column('Username', db.String(50), nullable=False, unique=True)
    email = db.Column('Email', db.String(100), nullable=False, unique=True)
    password = db.Column('Password', db.String(100), nullable=False)
    created_at = db.Column('CreatedAt', db.DateTime, default=datetime.utcnow)

    # Relationships
    tasks_created = db.relationship('Task', backref='creator', lazy=True, foreign_keys='Task.created_by')
    tasks_assigned = db.relationship('Task', backref='assignee', lazy=True, foreign_keys='Task.assigned_to')
    comments = db.relationship('Comment', backref='user', lazy=True)

class Task(db.Model):
    __tablename__ = 'Tasks'

    id = db.Column('TaskID', db.Integer, primary_key=True)
    title = db.Column('Title', db.String(100), nullable=False)
    description = db.Column('Description', db.Text)
    status = db.Column('Status', db.Enum('Pending', 'In Progress', 'Completed'), nullable=False, default='Pending')
    priority = db.Column('Priority', db.Enum('Low', 'Medium', 'High'), nullable=False, default='Medium')
    category = db.Column('Category', db.String(100))
    assigned_to = db.Column('AssignedTo', db.Integer, db.ForeignKey('Users.UserID'))
    created_by = db.Column('CreatedBy', db.Integer, db.ForeignKey('Users.UserID'), nullable=False)
    due_date = db.Column('DueDate', db.Date)
    created_at = db.Column('CreatedAt', db.DateTime, default=datetime.utcnow)

    comments = db.relationship('Comment', backref='task', lazy=True)

class Comment(db.Model):
    __tablename__ = 'Comments'

    id = db.Column('CommentID', db.Integer, primary_key=True)
    task_id = db.Column('TaskID', db.Integer, db.ForeignKey('Tasks.TaskID'), nullable=False)
    comment_text = db.Column('Comment', db.Text)
```

```
    user_id = db.Column('UserID', db.Integer, db.ForeignKey('Users.UserID'))
    created_at = db.Column('CreatedAt', db.DateTime, default=datetime.utcnow)


class Login(db.Model, UserMixin):
    __tablename__ = 'login'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(255), unique=True, nullable=False)
    email = db.Column(db.String(255))
    password_hash = db.Column(db.String(255), nullable=False)

    def set_password(self, password):
        self.password_hash = bcrypt.generate_password_hash(password)

    def check_password(self, password):
        return bcrypt.check_password_hash(self.password_hash, password)

    def __repr__(self):
        return f'<Login {self.username}>'


with app.app_context():
    db.create_all()
```

/app/auth.py

```
from app import login_manager
from .models import Login

@login_manager.user_loader
def load_user(user_id):
    return Login.query.get(int(user_id))
```

/app/__init__.py:

```
import os
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
```

```
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
from dotenv import load_dotenv


load_dotenv()

app = Flask(__name__)
app.config["SECRET_KEY"] = os.getenv("SECRET_KEY")
app.config["SQLALCHEMY_DATABASE_URI"] = os.getenv("SQLALCHEMY_DATABASE_URI")
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)


login_manager = LoginManager()
login_manager.init_app(app)

from app import routes, models, auth
```

run.py:

```
from app import app

if __name__ == "__main__":
    app.run(debug=True)
```

/app/templates/base.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Management System</title>
    <!-- Bootstrap CDN -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <!-- Semantic UI CDN -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.1/semantic.min.css">
```

```
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">TaskManager</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
        </li>
      </li class="nav-item active">
        <a class="nav-link" href="/users">Users</a>
      </li>
      </ul>
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          {% if current_user.is_authenticated %}
          <a class="btn btn-primary" href="/logout">Logout</a>
          {% else %}
          <a class="btn btn-primary" href="/login">Login/Signup</a>
          {% endif %}
        </li>
      </ul>
    </div>
  </nav>
  <div class="container mt-4">
    {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
      {% for category, message in messages %}
        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
          {{ message }}
          <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
      {% endfor %}
    {% endif %}
```

```
    {% endwith %}

    {% block content %}{% endblock %}
  </div>

  <!-- Bootstrap and jQuery Scripts -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.9/dist/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

/app/templates/create_task.html:

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
  <div class="row justify-content-center">
    <div class="col-lg-6 col-md-8 col-sm-12">
      <h2 class="text-center">Create Task</h2>
      <form action="/create_task" method="POST">
        <div class="form-group">
          <label for="title">Title</label>
          <input type="text" class="form-control" id="title" name="title" required>
        </div>
        <div class="form-group">
          <label for="description">Description</label>
          <textarea class="form-control" id="description" name="description"></textarea>
        </div>
        <div class="form-group">
          <label for="status">Status</label>
          <select class="form-control" id="status" name="status">
            <option>Pending</option>
            <option>In Progress</option>
            <option>Completed</option>
          </select>
```

```
            </div>
            <div class="form-group">
                <label for="priority">Priority</label>
                <select class="form-control" id="priority" name="priority">
                    <option>Low</option>
                    <option>Medium</option>
                    <option>High</option>
                </select>
            </div>
            <div class="form-group">
                <label for="category">Category</label>
                <input type="text" class="form-control" id="category" name="category">
            </div>
            <div class="form-group">
                <label for="assignedTo">Assigned To</label>
                <select class="form-control" id="assignedTo" name="assignedTo">
                    {% for user in users %}
                    <option value="{{ user.id }}">{{ user.username }}</option>
                    {% endfor %}
                </select>
            </div>
            <div class="form-group">
                <label for="createdBy">Created By</label>
                <select class="form-control" id="createdBy" name="createdBy">
                    {% for user in users %}
                    <option value="{{ user.id }}">{{ user.username }}</option>
                    {% endfor %}
                </select>
            </div>
            <div class="form-group">
                <label for="dueDate">Due Date</label>
                <input type="date" class="form-control" id="dueDate" name="dueDate">
            </div>
            <div class="text-center">
                <button type="submit" class="btn btn-primary">Create Task</button>
            </div>
        </form>
    </div>
  </div>
</div>
```

```
{% endblock %}
```

/app/templates/create_user.html:

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
    <div class="row justify-content-center">
        <div class="col-lg-6 col-md-8 col-sm-12">
            <h2 class="text-center">Create User</h2>
            <form action="/create_user" method="POST">
                <div class="form-group">
                    <label for="username">Username</label>
                    <input type="text" class="form-control" id="username" name="username" required>
                </div>
                <div class="form-group">
                    <label for="email">Email address</label>
                    <input type="email" class="form-control" id="email" name="email" required>
                </div>
                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" class="form-control" id="password" name="password" required>
                </div>
                <div class="text-center">
                    <button type="submit" class="btn btn-primary">Submit</button>
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

/app/templates/dashboard.html:

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
    <h2>Task Dashboard</h2>
    <a href="/create_task" class="btn btn-success mb-3">Create New Task</a>
    <div class="mb-3">
        <form method="GET" action="/dashboard">
            <div class="row">
                <div class="col">
                    <select class="form-control" name="user">
                        <option value="all">All Users</option>
                        {% for user in users %}
                        <option value="{{ user.id }}" {% if user.username == user_filter %}selected{% endif %}>{{
user.username }}</option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col">
                    <select class="form-control" name="priority">
                        <option value="all">All Priorities</option>
                        <option value="Low" {% if priority_filter == 'Low' %}selected{% endif %}>Low</option>
                        <option value="Medium" {% if priority_filter == 'Medium' %}selected{% endif %}>Medium</option>
                        <option value="High" {% if priority_filter == 'High' %}selected{% endif %}>High</option>
                    </select>
                </div>
                <div class="col">
                    <button type="submit" class="btn btn-primary">Apply Filter</button>
                </div>
            </div>
        </form>
    </div>
    <div class="row">
        {% set statuses = ['Pending', 'In Progress', 'Completed'] %}
        {% for status in statuses %}
        <div class="col-md-4 border">
            <h4 class="text-center mt-2">{{ status }}</h4>
            <div class="d-flex flex-column">
                {% for task_data in tasks %}
                {% set task = task_data[0] %}
```

```
            {% set assigned_username = task_data[-1] %}
            {% if task.status == status %}
            <a href="/task_details/{{ task.id }}" class="text-decoration-none text-dark">
                <div class="card m-2 border">
                    <div class="card-body">
                        <h5 class="card-title">{{ task.title }}</h5>
                        <h6 class="card-subtitle mb-2 text-muted">Priority: {{ task.priority }}</h6>
                        <p class="card-text">{{ task.description }}</p>
                        <p class="card-text"><small>Assigned To: {{ assigned_username }}</small></p>
                    </div>
                </div>
            </a>
            {% endif %}
            {% endfor %}
        </div>
    </div>
    {% endfor %}
  </div>
</div>
{% endblock %}
```

/app/templates/edit_task.html:

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
  <h2>Edit Task: {{ task.title }}</h2>
  <form action="/edit_task/{{ task.id }}" method="POST">
    <div class="form-group">
      <label for="title">Title</label>
      <input type="text" class="form-control" id="title" name="title" value="{{ task.title }}" required>
    </div>
    <div class="form-group">
      <label for="description">Description</label>
      <textarea class="form-control" id="description" name="description">{{ task.description }}</textarea>
    </div>
```

```
        <div class="form-group">
            <label for="status">Status</label>
            <select class="form-control" id="status" name="status">
                <option value="Pending" {% if task.status == 'Pending' %} selected {% endif %}>Pending</option>
                <option value="In Progress" {% if task.status == 'In Progress' %} selected {% endif %}>In
Progress</option>
                <option value="Completed" {% if task.status == 'Completed' %} selected {% endif
%}>Completed</option>
            </select>
        </div>
        <div class="form-group">
            <label for="priority">Priority</label>
            <select class="form-control" id="priority" name="priority">
                <option value="Low" {% if task.priority == 'Low' %} selected {% endif %}>Low</option>
                <option value="Medium" {% if task.priority == 'Medium' %} selected {% endif %}>Medium</option>
                <option value="High" {% if task.priority == 'High' %} selected {% endif %}>High</option>
            </select>
        </div>
        <div class="form-group">
            <label for="category">Category</label>
            <input type="text" class="form-control" id="category" name="category" value="{{ task.category }}">
        </div>
        <div class="form-group">
            <label for="assignedTo">Assigned To</label>
            <select class="form-control" id="assignedTo" name="assignedTo">
                {% for user in users %}
                <option value="{{ user.id }}" {% if task.assigned_to == user.id %} selected {% endif %}>{{
user.username }}</option>
                {% endfor %}
            </select>
        </div>

        <button type="submit" class="btn btn-primary">Update Task</button>
    </form>
</div>
{% endblock %}
```

/app/templates/index.html:

```
{% extends "base.html" %}

{% block content %}
<style>
    .centered-content {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 80vh; /* Adjust height as needed */
        text-align: center;
    }

    .background {
        background-image: url('your-background-image-url.jpg'); /* Add your background image URL here */
        background-size: cover;
        background-position: center;
    }

    .animated-element {
        animation: fadeIn 2s;
    }

    @keyframes fadeIn {
        from { opacity: 0; }
        to { opacity: 1; }
    }
</style>

<div class="background">
    <div class="centered-content">
        <div class="animated-element">
            <h1 class="display-4">Welcome to TaskManager!</h1>
            <p class="lead">An efficient solution to manage and track your tasks.</p>
            <hr class="my-4">
            <p>Assign tasks, set deadlines, update statuses, and collaborate effectively.</p>
            <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
        </div>
    </div>
</div>
{% endblock %}
```

/app/templates/login.html:

```
{% extends "base.html" %}

{% block content %}
<style>
    .auth-form {
        background: #f7f7f7;
        padding: 40px;
        border-radius: 10px;
        box-shadow: 0 4px 8px rgba(0,0,0,0.1);
        max-width: 400px;
        margin: 50px auto;
    }
    .form-control {
        border-radius: 20px;
    }
    .auth-button {
        background-color: #4CAF50;
        color: white;
        border-radius: 20px;
        padding: 10px 20px;
        font-size: 16px;
    }
    .auth-button:hover {
        background-color: #45a049;
    }
</style>

<div class="auth-form">
    <h2 class="text-center">Login</h2>
    <form method="POST" action="/login">
        <div class="form-group">
            <input type="text" class="form-control" name="username" placeholder="Email" required>
        </div>
        <div class="form-group">
```

```
            <input type="password" class="form-control" name="password" placeholder="Password" required>
        </div>
        <button type="submit" class="btn auth-button w-100">Login</button>
    </form>
    <p class="text-center mt-3">
        Don't have an account? <a href="/signup">Sign up</a>
    </p>
</div>
{% endblock %}
```

/app/templates/signup.html:

```
{% extends "base.html" %}

{% block content %}
<style>
    .auth-form {
        background: #fff6f6;
        padding: 40px;
        border-radius: 10px;
        box-shadow: 0 4px 8px rgba(0,0,0,0.1);
        max-width: 400px;
        margin: 50px auto;
    }
    .form-control {
        border-radius: 20px;
    }
    .auth-button {
        background-color: #FF6347;
        color: white;
        border-radius: 20px;
        padding: 10px 20px;
        font-size: 16px;
    }
    .auth-button:hover {
        background-color: #ff5733;
    }
```

```
</style>

<div class="auth-form">
    <h2 class="text-center">Sign Up</h2>
    <form method="POST" action="/signup">
        <div class="form-group">
            <input type="text" class="form-control" name="username" placeholder="Username" required>
        </div>
        <div class="form-group">
            <input type="email" class="form-control" name="email" placeholder="Email" required>
        </div>
        <div class="form-group">
            <input type="password" class="form-control" name="password" placeholder="Password" required>
        </div>
        <button type="submit" class="btn auth-button w-100">Sign Up</button>
    </form>
    <p class="text-center mt-3">
        Already have an account? <a href="/login">Login</a>
    </p>
</div>
{% endblock %}
```

/app/templates/task_details.html:

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
    <h2>Task Details: {{ task.title }}</h2>
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">{{ task.title }}</h5>
            <p class="card-text">{{ task.description }}</p>
            <p>Status: {{ task.status }}</p>
            <p>Priority: {{ task.priority }}</p>
            <p>Category: {{ task.category }}</p>
            <p>Assigned To: {{ task.assigned_username }}</p>
```

```
            <p>Due Date: {{ task.due_date }}</p>
            <a href="/edit_task/{{ task.id }}" class="btn btn-primary">Edit Task</a>
            <a href="/delete_task/{{ task.id }}" class="btn btn-danger">Delete Task</a>
        </div>
    </div>

    <div class="mt-4">
        <h3>Comments</h3>
        {% for comment in comments %}
        <div class="card mt-2">
            <div class="card-body">
                <strong>{{ comment.user.username }}:</strong>
                <p>{{ comment.comment_text }}</p>
                <a href="/delete_comment/{{ comment.id }}" class="btn btn-danger">Delete</a>
            </div>
        </div>
        {% endfor %}
        <form action="/add_comment/{{ task.id }}" method="POST" class="mt-3">
            <div class="form-group">
                <label for="user">Comment as:</label>
                <select class="form-control" id="user" name="user_id">
                    {% for user in users %}
                    <option value="{{ user.id }}">{{ user.username }}</option>
                    {% endfor %}
                </select>
            </div>
            <div class="form-group">
                <label for="comment">Comment:</label>
                <textarea class="form-control" id="comment" name="comment" placeholder="Add a
comment..."></textarea>
            </div>
            <button type="submit" class="btn btn-success">Add Comment</button>
        </form>
    </div>
</div>
{% endblock %}
```

/app/templates/users.html:

```
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
    <h2 style="text-align: center;">Users</h2>
    <a href="/create_user" class="btn btn-primary float-right mb-3">Create User</a>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">Username</th>
                <th scope="col">Email</th>
                <th scope="col">Created At</th>
                <th scope="col">Action</th>
            </tr>
        </thead>
        <tbody>
            {% for user in users %}
            <tr>
                <td>{{ user.username }}</td>
                <td>{{ user.email }}</td>
                <td>{{user.created_at}}</td>
                <td>
                    <form action="/delete_user/{{ user.id }}" method="POST">
                        <button type="submit" class="btn btn-danger">Delete</button>
                    </form>
                </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</div>
{% endblock %}
```