<u>MoodMinder - Mood & Task Tracker</u>

By

Prathamesh Prakash Patil – AF04956000

Index

Acknowledgement

The project **"Online News Portal with Sentiment Analysis"** is the Project work carried out by

Name	Enrollment No
Prathamesh Prakash Patil	**AF04956000**

Under the Guidance.

We are thankful to my project guide for guiding me to complete the Project.

His suggestions and valuable information regarding the formation of the Project

Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

<u>Abstract</u>

The project called **MoodMinder - Mood & Task Tracker** is about creating a website to share news quickly, safely, and in an organized way. This system replaces the old manual way of publishing news with a digital platform where administrators can easily add, edit, delete, and manage news articles. It also stores news for a long time, making it easier and faster to handle.

This portal lets users browse news by category, read detailed articles with publication details, and share their thoughts through comments without needing to log in. It also includes a feature to analyze comments, showing whether they are positive, neutral, or negative. This helps administrators understand public feedback better.

The system is simple, easy to use, and works well on mobile devices. It allows real-time updates, comment moderation, and better analysis. It helps organizations save time and resources while reaching more people. This platform focuses on engagement, accessibility, and making decisions based on audience feedback.

1. Introduction

The MoodMinder (Mood and Task Tracker) is a full-stack web application designed to help users keep track of their daily activities, monitor their moods, and analyze their energy levels over time. In today's fast-paced lifestyle, individuals often struggle to balance productivity with mental well-being. This system bridges that gap by providing a simple yet effective way to log tasks, track the time spent, and record emotional states along with energy levels.

The application not only allows users to maintain logs of their daily tasks but also provides visual analytics through charts, offering insights into patterns of mood and energy fluctuations. These insights help users identify habits, lifestyle choices, or activities that positively or negatively impact their mental health and productivity.

Additionally, the system generates personalized suggestions based on the user's latest mood and energy level to promote mental well-being. Features like user authentication, profile management, report generation (in PDF format), and data filtering make it a complete solution for managing both productivity and wellness.

The MoodMinder is built using React (frontend), Node.js with Express.js (backend), and MySQL (database), ensuring a scalable and efficient architecture. The clean UI, powered by Bootstrap and React, provides users with an engaging and user-friendly experience.

1.1 Objective of the Present Work

The main objectives of the MoodMinder (Mood and Task Tracker) project are as follows:

1. Task Logging – To allow users to record their daily tasks along with time spent, providing a structured overview of productivity.

2. Mood & Energy Tracking – To help users monitor their emotional state and energy level for each activity, creating a personal wellness record.

3. Data Analytics – To provide meaningful insights using charts and visualizations that highlight mood and energy patterns over time.

4. Personalized Suggestions – To generate simple and practical suggestions based on user's latest mood and energy data for improving mental health and productivity.

5. User Authentication & Profile Management – To ensure secure access by implementing signup, login, and profile editing features.

6. Report Generation – To enable users to export their activity logs and analytics in a professional PDF report format for future reference.

7. Filtering & Searching – To allow users to filter logs based on date ranges and view specific activity details efficiently.

8. User-Friendly Interface – To provide an easy-to-use and visually appealing interface that encourages consistent usage.

<u>System analysis</u>

3.1 PROBLEM DEFINITION

In today's fast-paced lifestyle, people often struggle to keep track of their daily activities, productivity levels, and emotional well-being. While many productivity tools exist, most of them only focus on task management without considering the mental and emotional state of the user.

Lack of awareness about one's mood patterns and energy fluctuations often leads to stress, burnout, and reduced efficiency. Similarly, the absence of an integrated platform that combines task tracking, mood monitoring, and actionable insights creates a gap for individuals who want to balance both productivity and mental wellness.

Therefore, there is a need for a simple yet effective system that not only logs tasks but also tracks emotional and energy levels, analyzes them, and provides personalized suggestions for improvement.

3.2 Preliminary Investigation Purpose

The MoodMinder is designed to make news accessible anytime and anywhere through a digital platform. It aims to replace traditional news publishing methods by allowing administrators to manage news articles efficiently while providing users with real-time updates and interactive features like commenting and sentiment analysis.

Benefits

The portal provides several advantages:

- **Instant News Access**: Users can browse news articles categorized by topics such as politics, business, sports, and entertainment.
- **Efficient log Management**: Admins can easily add, edit, delete, and restore logs without delays.
- **User Engagement**: Readers can interact by posting comments without login
- **Improved Moderation**: Sentiment analysis helps admins monitor user feedback and moderate discussions effectively.

Proposed System

The proposed system offers a structured, reliable, and automated approach to news publishing:

- Users can browse articles, comment on news stories, and receive instant updates.

- Administrators can moderate comments, manage news categories, and analyze user sentiment.
- Sentiment analysis provides insights into reader reactions, making content management more effective.
- The portal is designed to be simple, responsive, and efficient, ensuring a smooth experience for both users and administrators.

3.3 Feasibility Study

The feasibility study evaluates whether the **MoodMinder – mood & task tracker** project is practical, achievable, and beneficial. This assessment ensures the system can be successfully implemented within available resources, technology, and constraints.

Types of Feasibility Analysis

- **Technical Feasibility**
  - ✔ The portal is built using **Node.js for backend** and **HTML, CSS, Bootstrap for frontend**, making it compatible with existing web technologies.
  - ✔ The database is managed using **MySQL**, ensuring secure and scalable data storage.
  - ✔ The system is hosted on a **reliable web server**, supporting high user traffic.

- **Economic Feasibility**
  - ✔ Open-source technologies like MySQL help minimize software costs.
  - ✔ The automated sentiment analysis feature saves administrative effort in assessing user feedback trends.

- **Operational Feasibility**
  - ✔ Users can easily access the portal through any device, ensuring high usability.
  - ✔ News articles are categorized, making navigation simple.
  - ✔ Admins can efficiently moderate comments, ensuring a safe and engaging space for readers.

- **Schedule Feasibility**
  - ✔ The development timeline is realistic, covering design, coding, testing, and deployment within estimated deadlines.

- ✔ Modular development ensures different parts of the system are built incrementally, improving flexibility.

3.4Project Planning

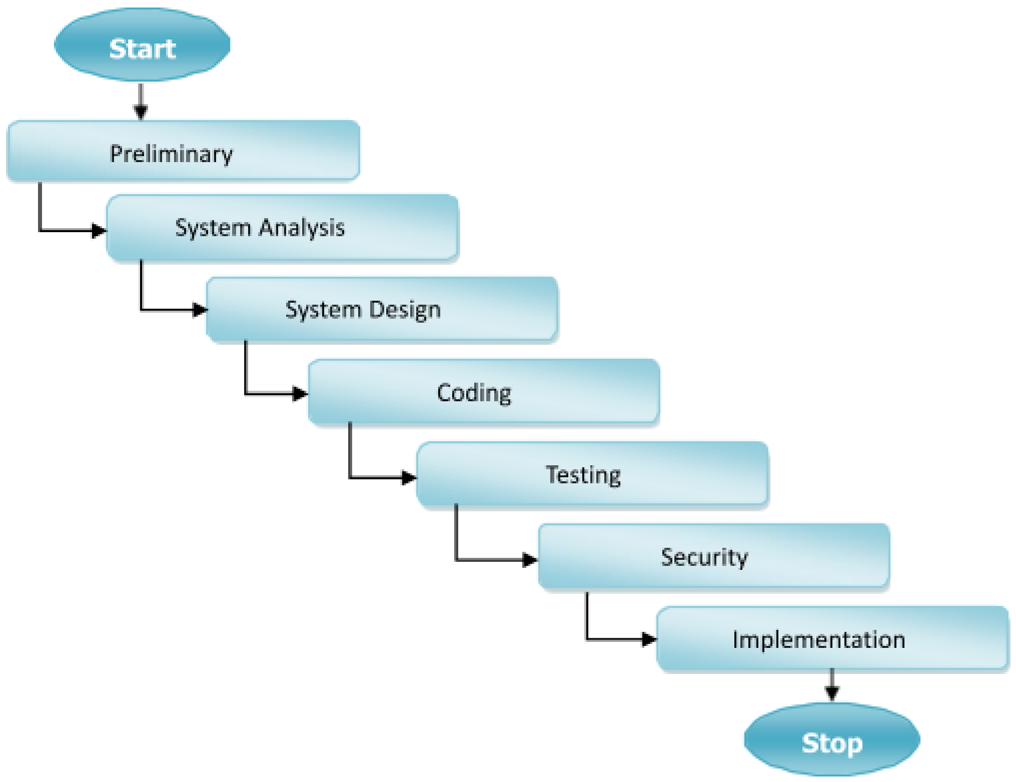
Purpose of Project Planning

Project planning ensures that the development of the **MoodMinder – mood & task tracker** follows a structured approach. It helps define the workflow, resource allocation, timelines, and risks involved to deliver the system efficiently.

Phases Covered in the Plan

The planning is divided into different phases to ensure smooth execution:

1. **Preliminary Investigation** – Understanding the project scope and objectives.
2. **System Analysis** – Identifying challenges, gathering requirements, and defining solutions.
3. **System Design** – Structuring modules, database design, and UI development.
4. **Coding** – Developing the portal using JavaScript (Node.js).
5. **Security** – Implementing authentication, data encryption, and user privacy measures.
6. **Testing** – Performing unit testing, integration testing, and user acceptance testing.
7. **Implementation** – Deploying the final system and ensuring smooth operation.



3.5 Software Requirement Specification (SRS)

The **Software Requirement Specification (SRS)** outlines the fundamental requirements of the **MoodMinder – mood & task tracker** to ensure efficient functionality, usability, and maintainability.

Software & Hardware Requirements Software Requirements

- **Frontend**: HTML, CSS, Bootstrap for responsive UI
- **Backend**: Node.js
- **Database**: MySQL for storing logs and user details

Hardware Requirements

- **Processor**: Intel i5 or higher
- **RAM**: Minimum 8GB
- **Storage**: At least 100GB for database and media files
- **Connectivity**: Internet access for real-time updates

3.6 Functional Requirements

1. User Module

Users can:

- Read news articles categorized into Politics, Business, Technology, Sports, Entertainment, etc.
- Search for specific articles using keywords.
- Post comments on news stories without requiring a login.

3.7 Software Engineering Paradigm

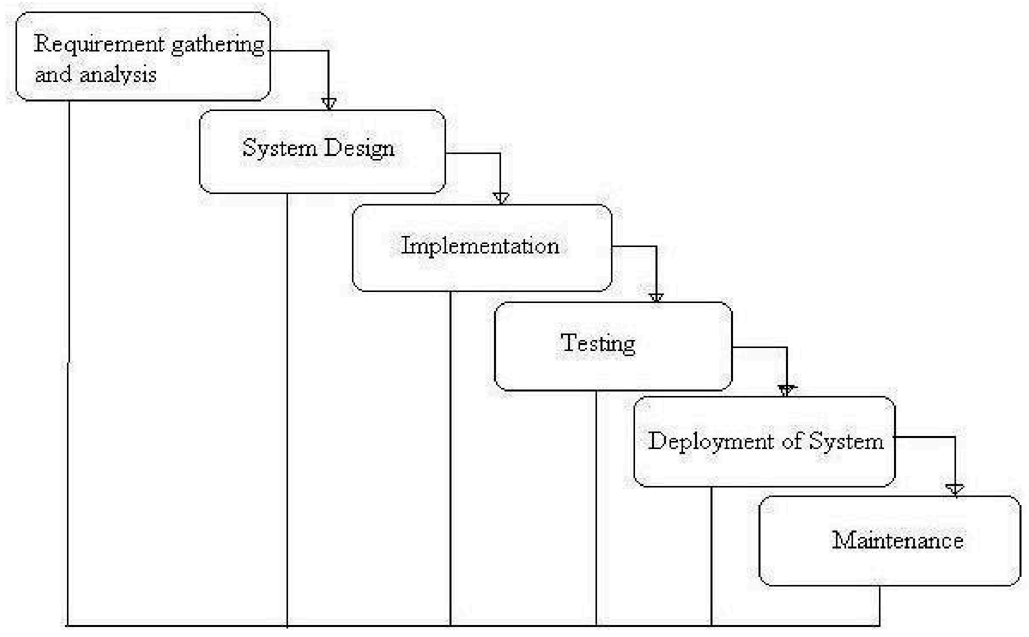
The development of the **MoodMinder – mood & task tracker** follows a structured approach to ensure efficiency, reliability, and maintainability. The chosen paradigm helps streamline the project by defining clear phases while allowing iterative improvements.

Development Model: Adapted Waterfall Model

The **Waterfall Model** is traditionally a linear approach, but for this project, an **iterative feedback mechanism** is incorporated. This helps refine earlier phases based on insights gathered during implementation.

Key Adaptations in the Waterfall Model:

1. **Structured Phase Progression** – Each stage follows a defined sequence, ensuring clarity in execution.
2. **Iterative Refinements** – Feedback loops allow adjustments, especially between testing and coding.
3. **Defined Milestones** – Each stage reaches completion before moving to the next phase.
4. **Flexible Adjustments** – Overlapping is permitted when necessary to enhance efficiency.



Phases of Development

1. **Requirement Analysis & System Study**
   - Identifying project goals, challenges, and functional specifications.
   - Gathering stakeholder requirements and defining core functionalities.
2. **System Design**
   - Structuring the **database, modules, and architecture**.
   - Designing **user interfaces** for optimal accessibility.
3. **Implementation (Coding)**
   - Backend development using **Mysql**.
   - Frontend design using **HTML, CSS, Bootstrap**.
   - Database integration with **MySQL**.
4. **Testing & Debugging**
   - Unit testing, integration testing, and usability checks.
   - Debugging for performance improvements.
5. **Deployment & Maintenance**
   - Hosting on a scalable environment.
   - Continuous updates for feature enhancements.

3.8 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

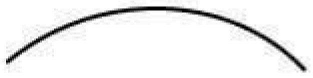
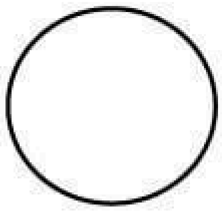
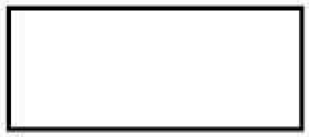
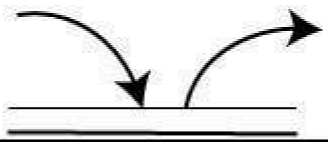
It shows how data enter and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:

Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

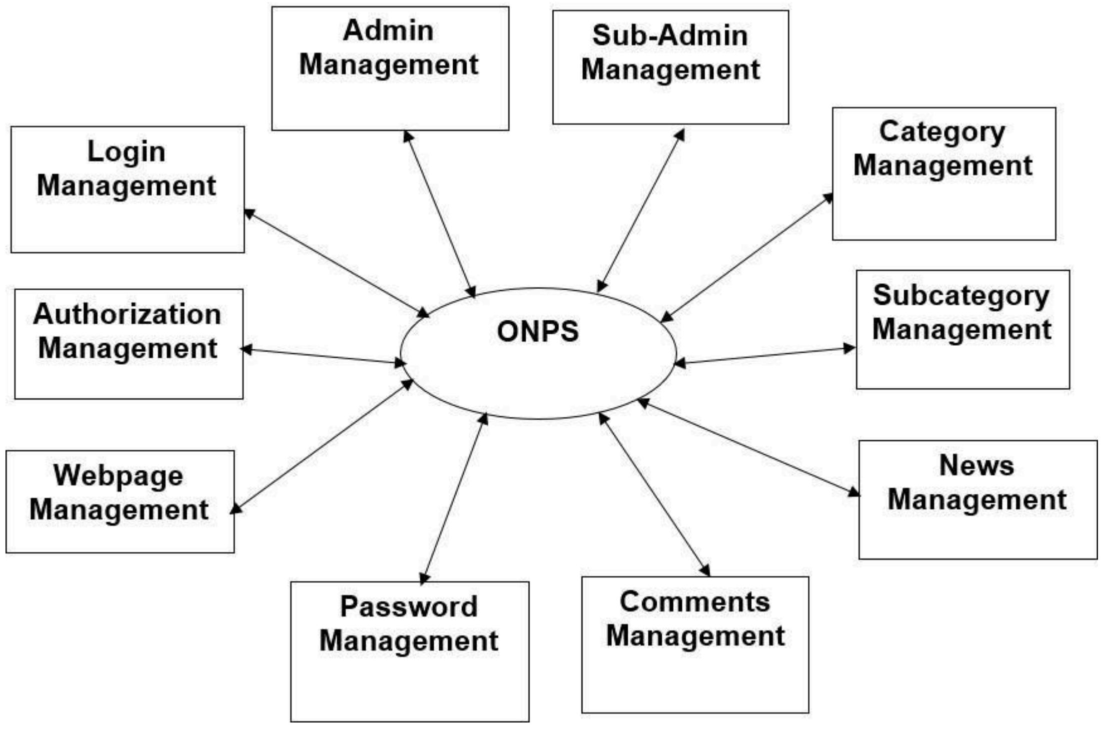
Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

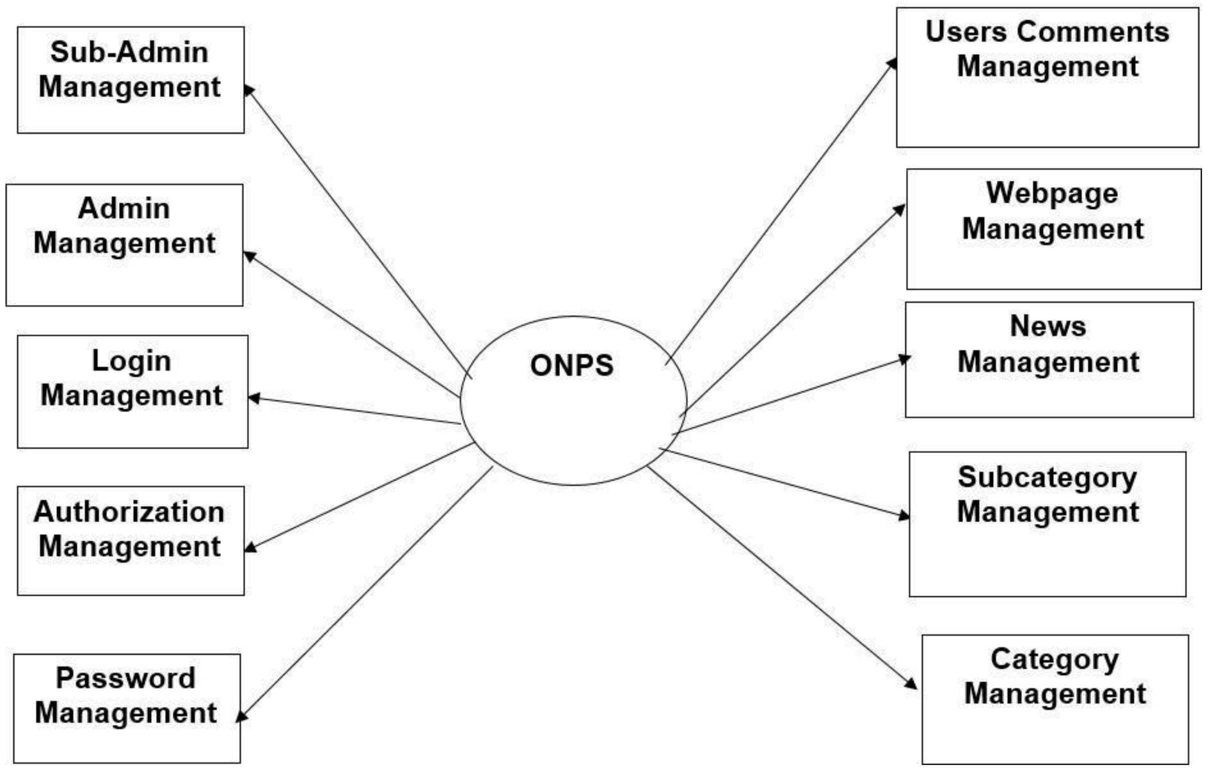
Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Zero Level DFD

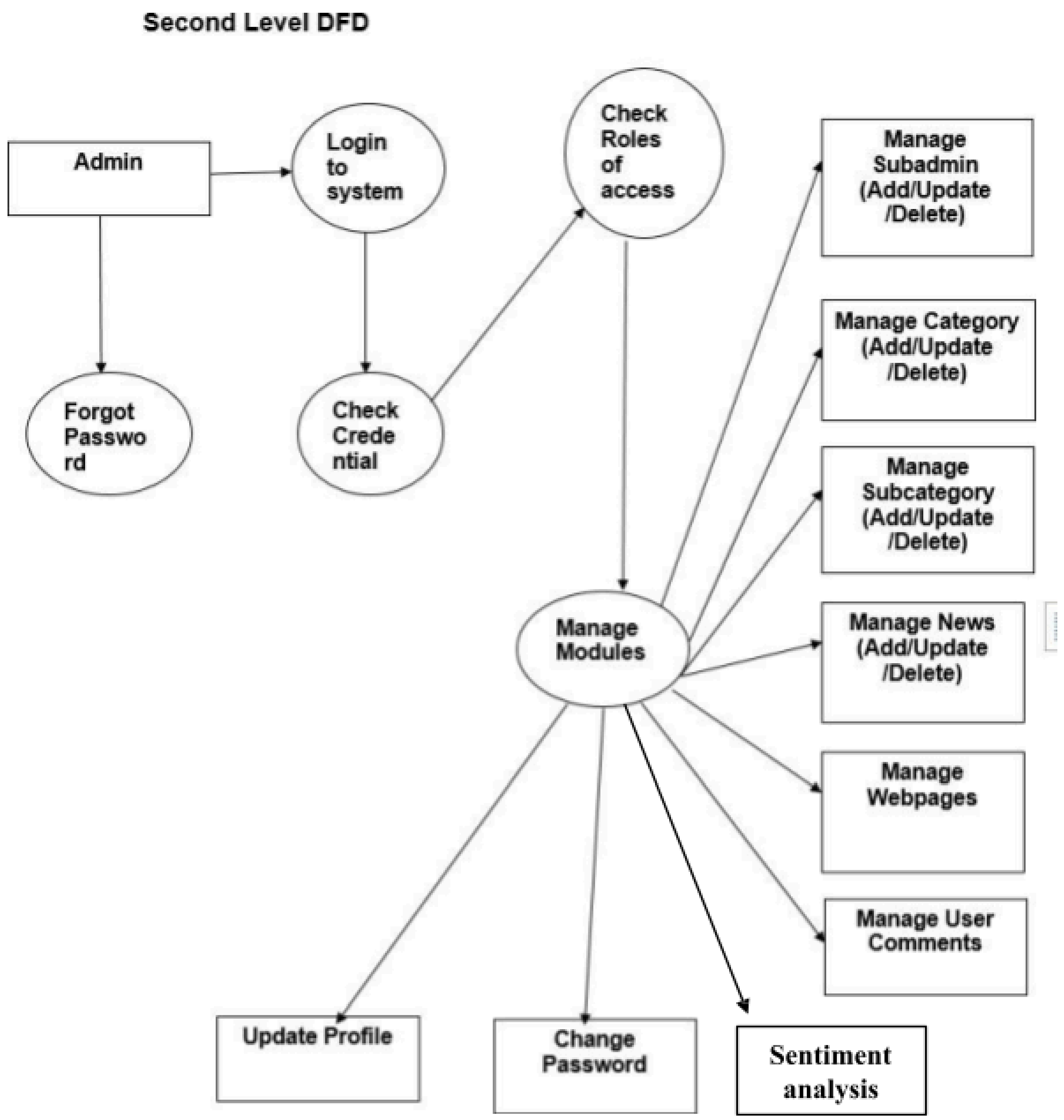


The **Zero Level DFD** of the MoodMinder – mood & task tracker shows the main processes interacting with the system. It highlights the flow of data between the central system and these modules, representing how each core function is managed within the portal.

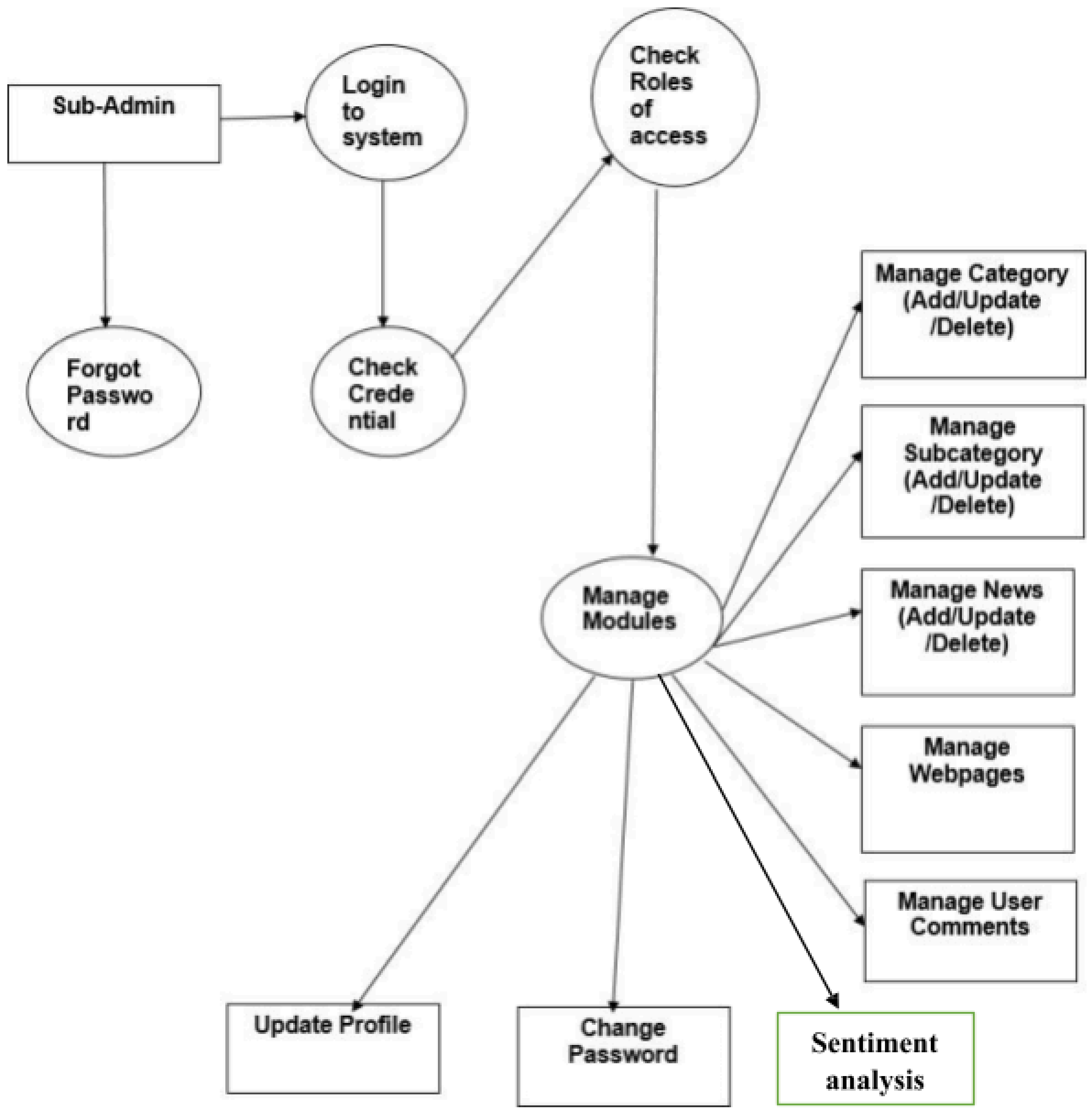
First Level DFD



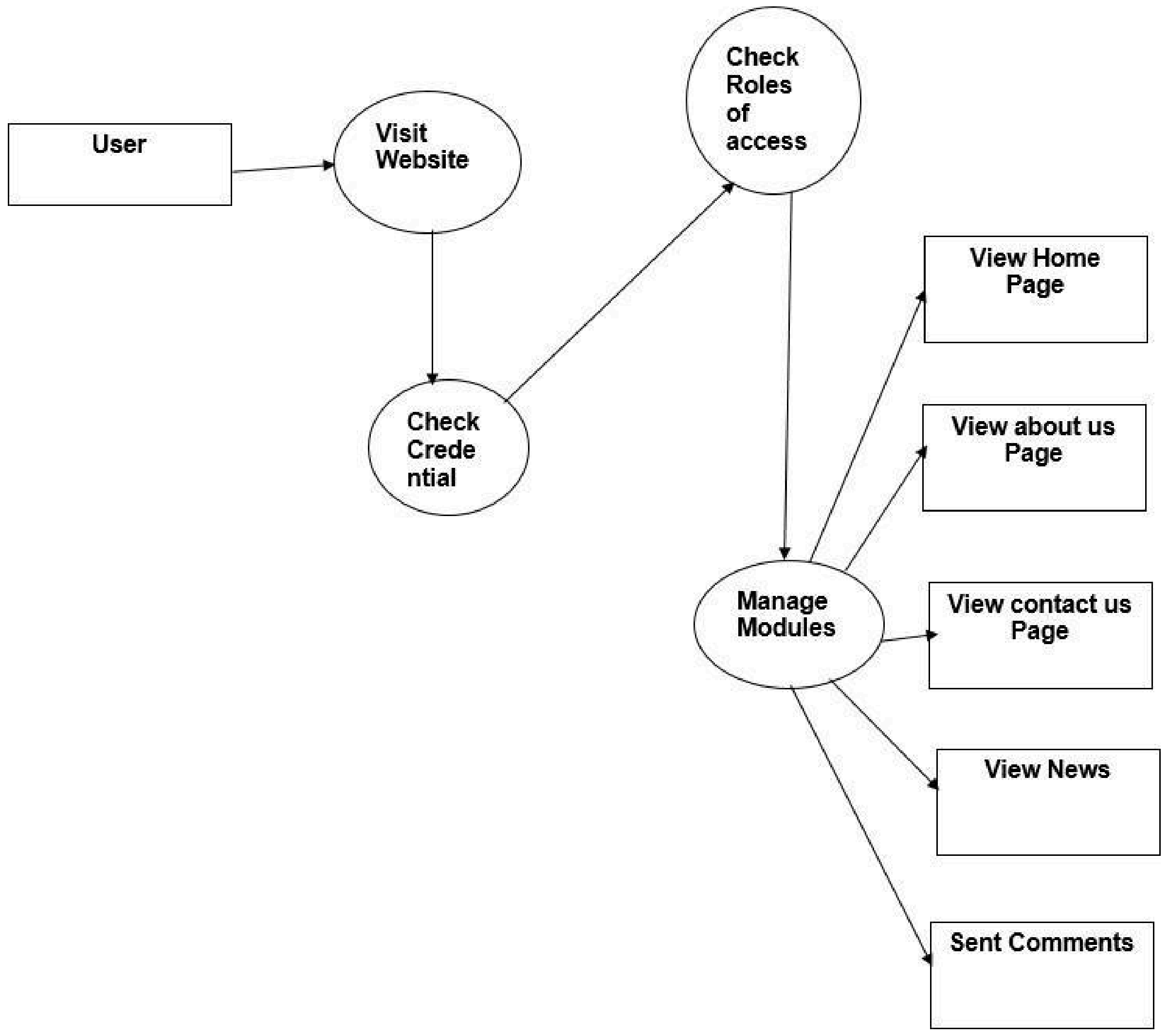
The **First-Level DFD** shows how the system handles key functions like login, authorization, and password management, and connects them to content-related modules such as category, subcategory, webpage, and user comments management for smooth portal operations.



The **Second-Level DFD** shows the internal working of the admin in the system. It includes processes like login, credential checking, role-based access, and module management. The admin can manage subadmins, categories, subcategories, news, webpages, and user comments, along with profile updates, password changes, and performing sentiment analysis.



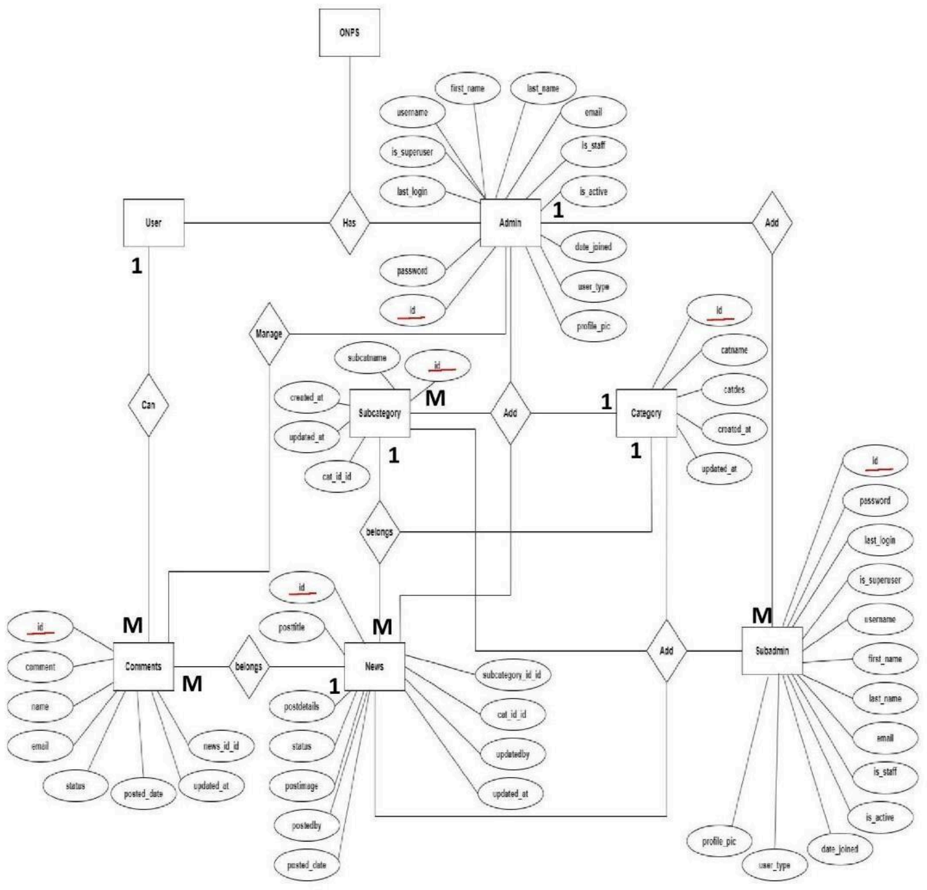
This **Second-Level DFD** represents the internal workflow for the Sub-Admin in the system. After logging in and verifying credentials, the system checks the Sub-Admin's access role and allows them to manage specific modules such as categories, subcategories, news, webpages, and user comments. Additionally, the Sub-Admin can update their profile, change their password, and perform sentiment analysis.



This **Second-Level DFD** illustrates the **User's interaction** with the system. When a user visits the website users can interact with various modules.

ER diagram

The **ER Diagram** of the **MoodMinder – mood & task tracker** shows key relationships between **Users and logs** ensuring structured data management and insightful moderation.



4. System design

4.1. Module

1. User Module

Handles signup, login, and authentication.

Stores user details like:

id (Primary Key)

name

email

password (encrypted)

Provides profile management (update name, email, password).

Responsible for generating authentication tokens (JWT).

2. Logs Module

Main module for task and mood tracking.

Each log entry is linked with a user (via user_id).

Stores details such as:

task_title

time_spent (minutes)

mood (happy, neutral, sad)

energy_level (low, medium, high)

notes

date

Provides CRUD operations:

Add new log

View logs (with date filters)

Edit log

Delete log

3. Analytics Module

Processes the logs and generates insights.

Uses charts (Pie chart for mood, Bar chart for energy).

Provides suggestions based on user's latest mood and energy level.

e.g., If mood is sad, suggest relaxation activities.

If energy is high, suggest focusing on important tasks.

4. Report Generation Module

Compiles user data and activity logs into a PDF report.

Includes:

User details

Logs summary

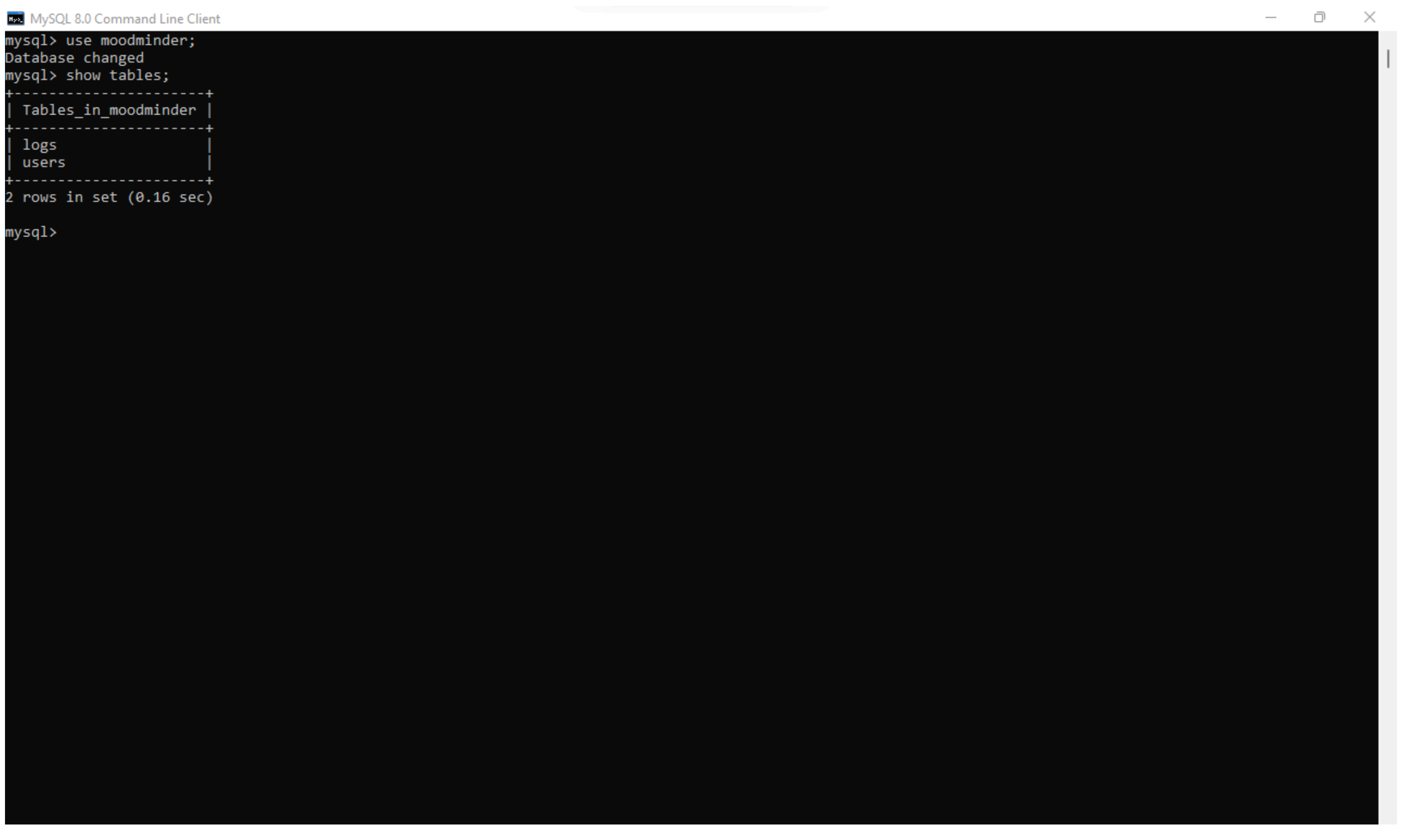
Suggestions based on latest log

Allows user to download the report for personal tracking.

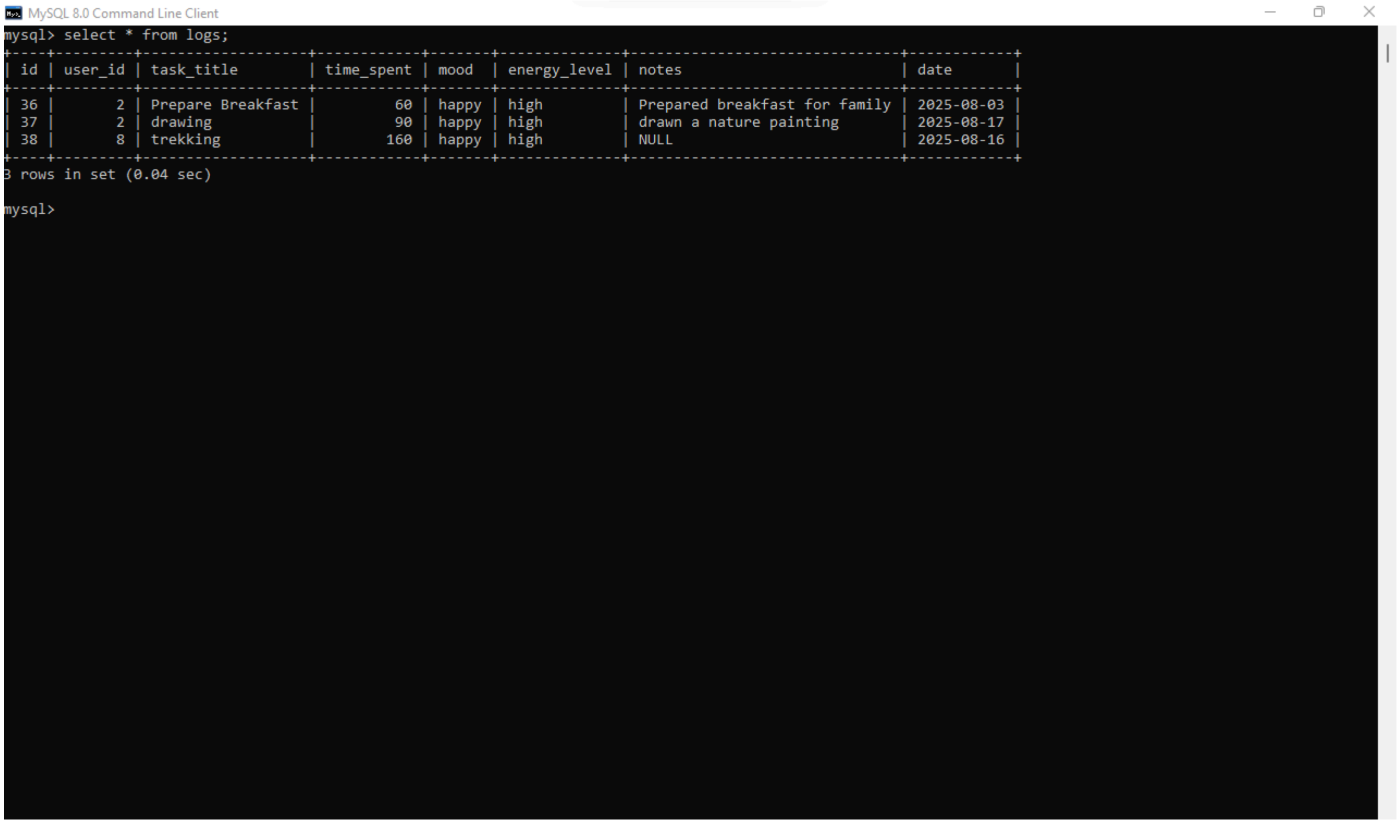
4.2 DATA STRUCTURE OF ALL MODULES:

We have organized one database **Moodminder** for system design. It can be accessed directly or sequentially by registered. The database determines files, record, fields, and characters. It can be easily controlled and updated.

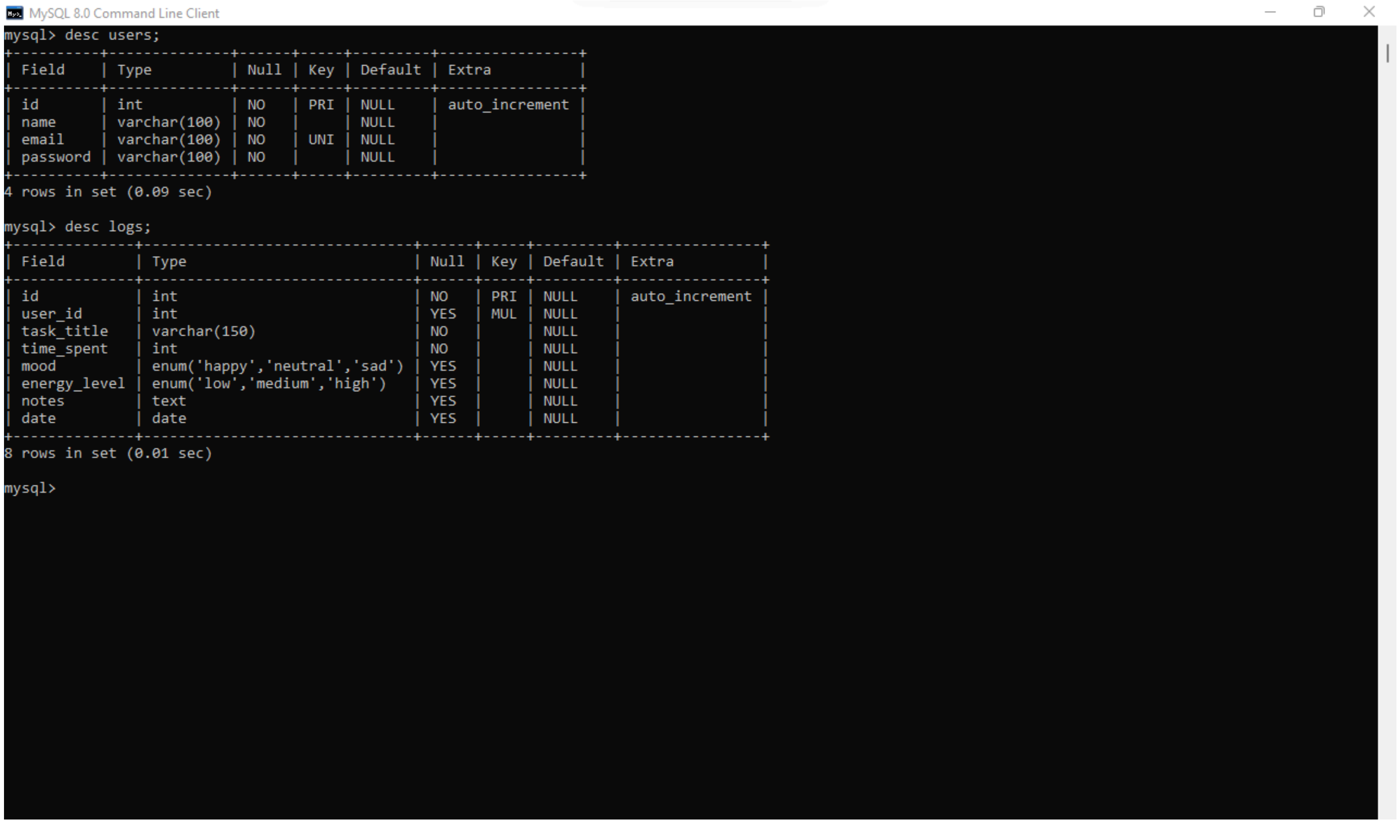
Customized Tables Details



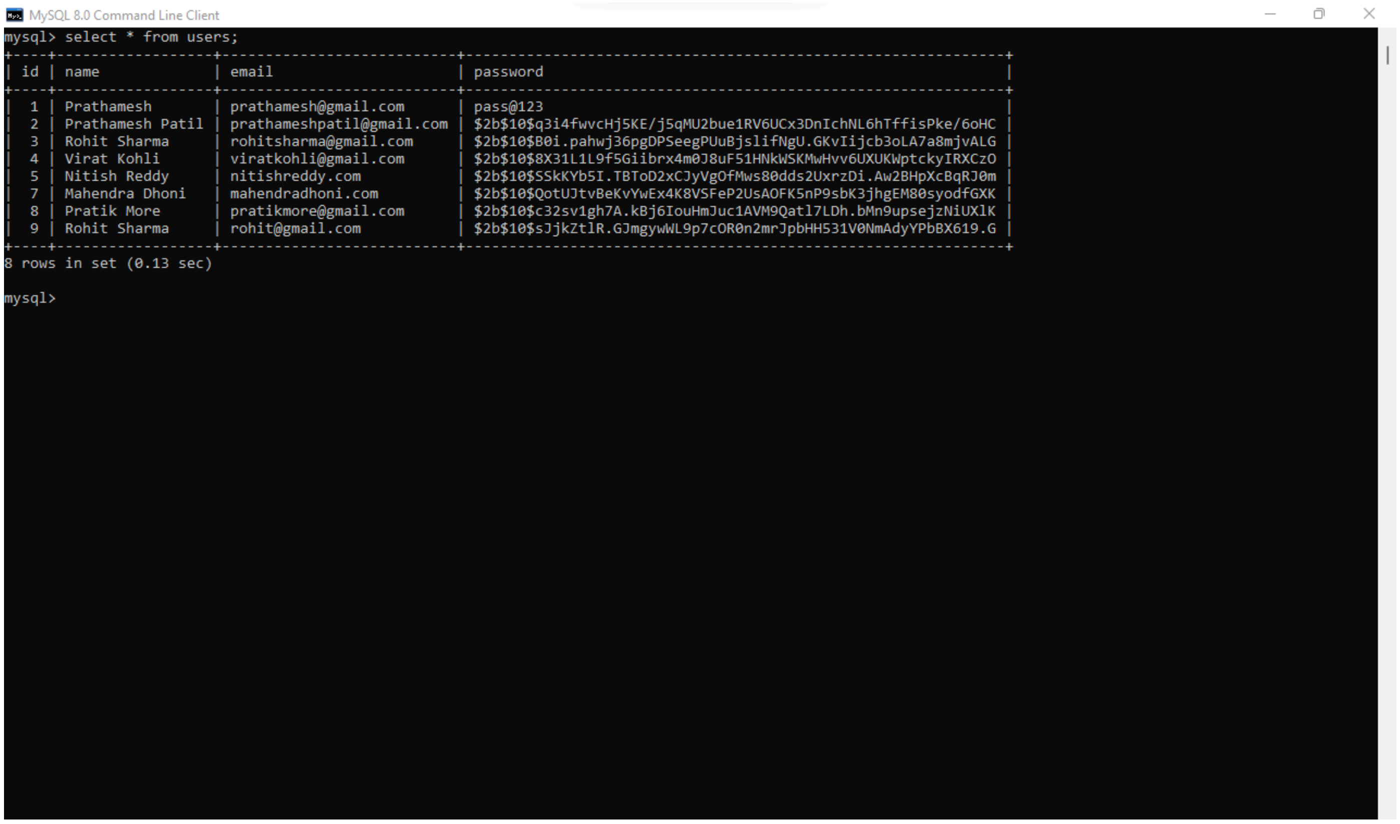
```
mysql> use moodminder;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_moodminder |
+----------------------+
| logs                 |
| users                |
+----------------------+
2 rows in set (0.16 sec)

mysql>
```



```
mysql> select * from logs;
+----+---------+-------------------+------------+-------+--------------+-------------------------------+------------+
| id | user_id | task_title        | time_spent | mood  | energy_level | notes                         | date       |
+----+---------+-------------------+------------+-------+--------------+-------------------------------+------------+
| 36 |       2 | Prepare Breakfast |         60 | happy | high         | Prepared breakfast for family | 2025-08-03 |
| 37 |       2 | drawing           |         90 | happy | high         | drawn a nature painting       | 2025-08-17 |
| 38 |       8 | trekking          |        160 | happy | high         | NULL                          | 2025-08-16 |
+----+---------+-------------------+------------+-------+--------------+-------------------------------+------------+
3 rows in set (0.04 sec)

mysql>
```



```
mysql> desc users;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| name     | varchar(100) | NO   |     | NULL    |                |
| email    | varchar(100) | NO   | UNI | NULL    |                |
| password | varchar(100) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
4 rows in set (0.09 sec)

mysql> desc logs;
+--------------+--------------------------------+------+-----+---------+----------------+
| Field        | Type                           | Null | Key | Default | Extra          |
+--------------+--------------------------------+------+-----+---------+----------------+
| id           | int                            | NO   | PRI | NULL    | auto_increment |
| user_id      | int                            | YES  | MUL | NULL    |                |
| task_title   | varchar(150)                   | NO   |     | NULL    |                |
| time_spent   | int                            | NO   |     | NULL    |                |
| mood         | enum('happy','neutral','sad')  | YES  |     | NULL    |                |
| energy_level | enum('low','medium','high')    | YES  |     | NULL    |                |
| notes        | text                           | YES  |     | NULL    |                |
| date         | date                           | YES  |     | NULL    |                |
+--------------+--------------------------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)

mysql>
```


MySQL 8.0 Command Line Client

```
mysql> select * from users;
+----+------------------+---------------------------+--------------------------------------------------------------+
| id | name             | email                     | password                                                     |
+----+------------------+---------------------------+--------------------------------------------------------------+
|  1 | Prathamesh       | prathamesh@gmail.com      | pass@123                                                     |
|  2 | Prathamesh Patil | prathameshpatil@gmail.com | $2b$10$q3i4fwvcHj5KE/j5qMU2bue1RV6UCx3DnIchNL6hTffisPke/6oHC |
|  3 | Rohit Sharma     | rohitsharma@gmail.com     | $2b$10$B0i.pahwj36pgDPSeegPUuBjslifNgU.GKvIijcb3oLA7a8mjvALG |
|  4 | Virat Kohli      | viratkohli@gmail.com      | $2b$10$8X31L1L9f5Giibrx4m0J8uF51HNkWSKMwHvv6UXUKWptckyIRXCzO |
|  5 | Nitish Reddy     | nitishreddy.com           | $2b$10$SSkKYb5I.TBToD2xCJyVgOfMws80dds2UxrzDi.Aw2BHpXcBqRJ0m |
|  7 | Mahendra Dhoni   | mahendradhoni.com         | $2b$10$QotUJtvBeKvYwEx4K8VSFeP2UsAOFK5nP9sbK3jhgEM80syodfGXK |
|  8 | Pratik More      | pratikmore@gmail.com      | $2b$10$c32sv1gh7A.kBj6IouHmJuc1AVM9Qatl7LDh.bMn9upsejzNiUXlK |
|  9 | Rohit Sharma     | rohit@gmail.com           | $2b$10$sJjkZtlR.GJmgywWL9p7cOR0n2mrJpbHH531V0NmAdyYPbBX619.G |
+----+------------------+---------------------------+--------------------------------------------------------------+
8 rows in set (0.13 sec)

mysql>
```

4.3 PROCEDURAL DESIGN:

Process logic (flowchart) of each module

4.3.1 User Panel Design

In user panel design we have done our task for user. Here we provide facility about Online News Paper. In index page user can select any options which is needed by him/her. By selecting options he/she can see the desired page. Then he/she can get the all oriented information finally. The design of user panel is shown in following flow chart….

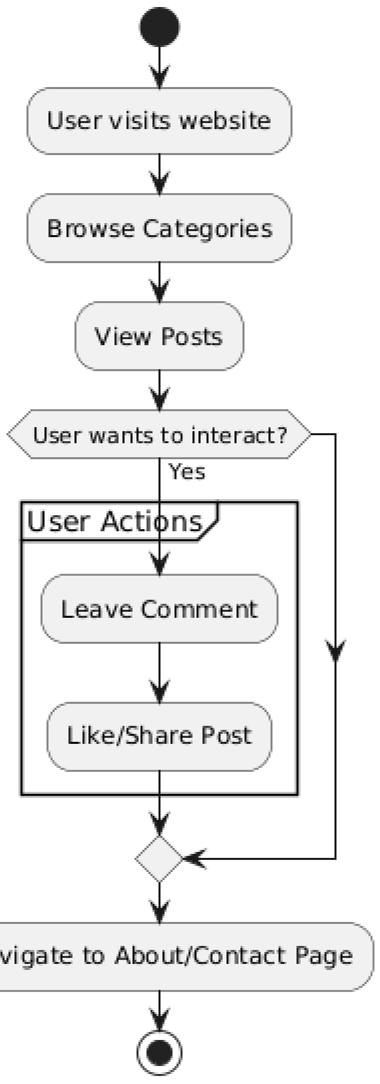
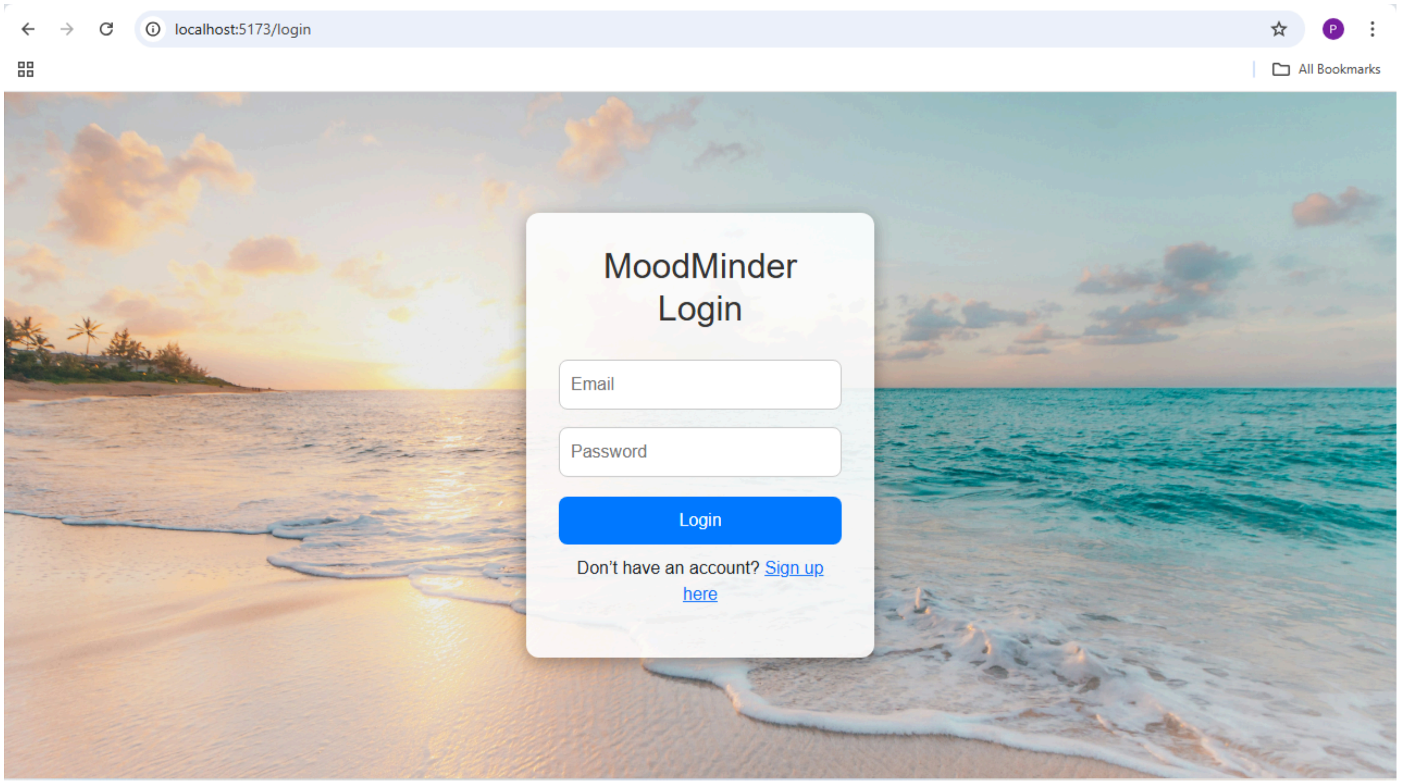
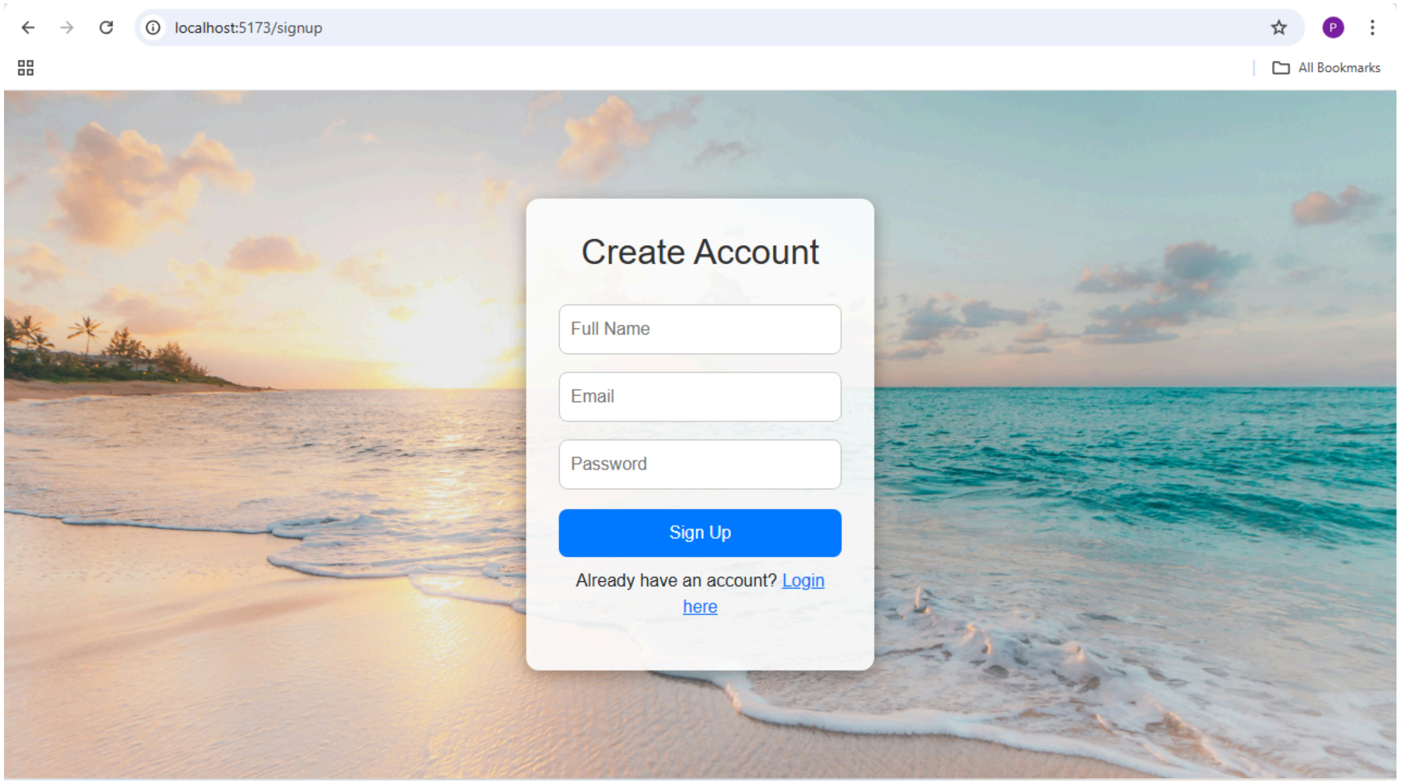
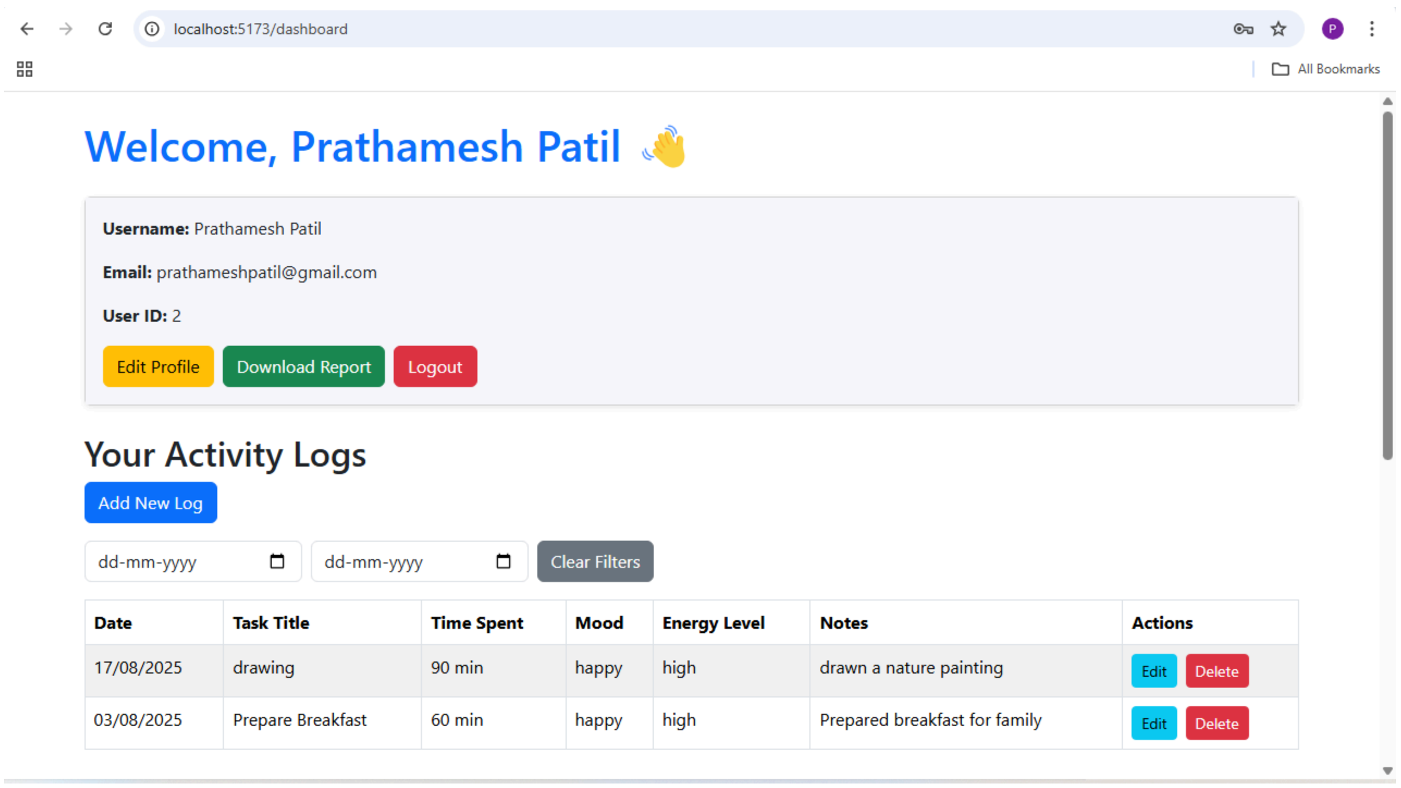


Fig. 4.1: The user panel flowchart part.

SCREENSHOTS







localhost:5173/dashboard

All Bookmarks

Welcome, Prathamesh Patil 👋

Username: Prathamesh Patil

Email: prathameshpatil@gmail.com

User ID: 2

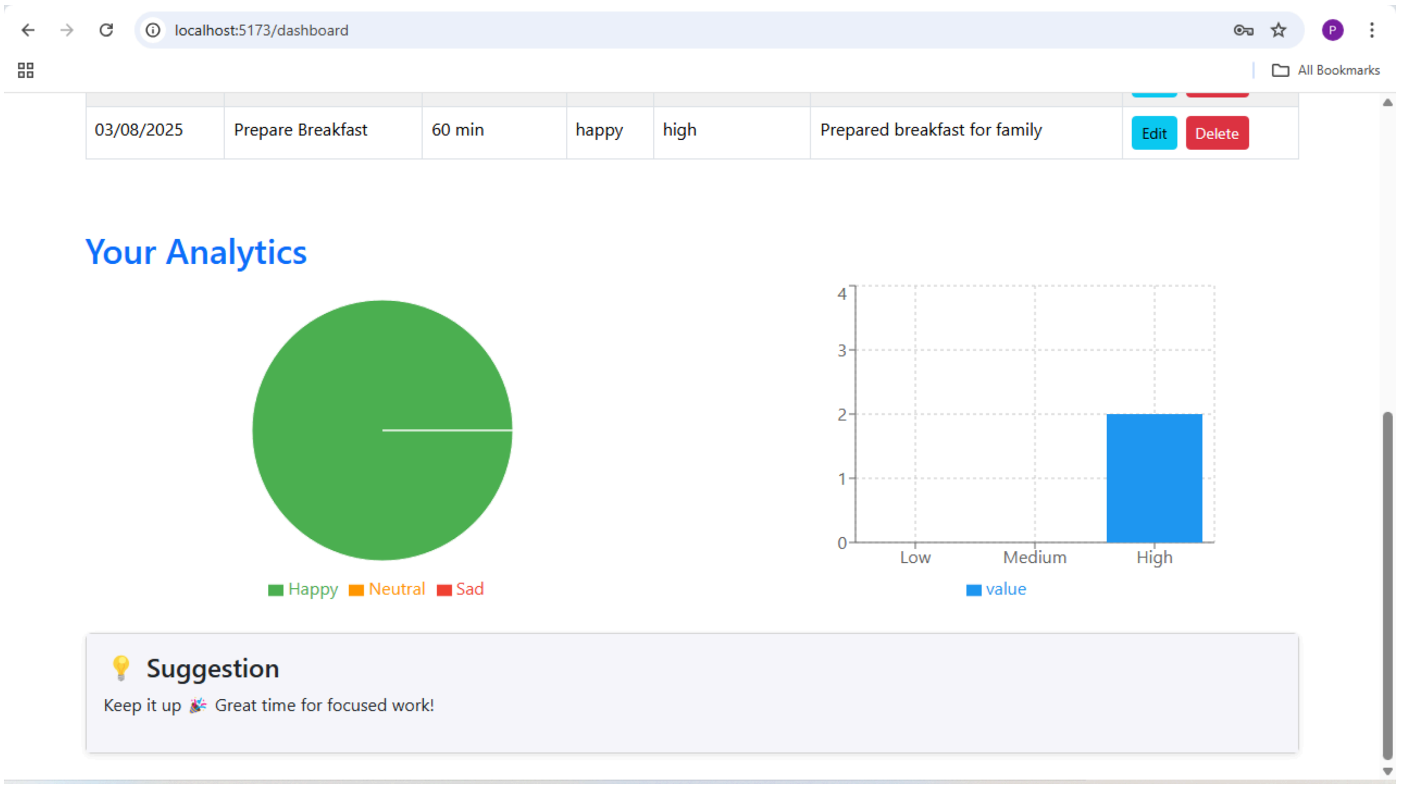
Edit Profile | Download Report | Logout

Your Activity Logs

Add New Log

dd-mm-yyyy | dd-mm-yyyy | Clear Filters

Date	Task Title	Time Spent	Mood	Energy Level	Notes	Actions
17/08/2025	drawing	90 min	happy	high	drawn a nature painting	Edit Delete
03/08/2025	Prepare Breakfast	60 min	happy	high	Prepared breakfast for family	Edit Delete

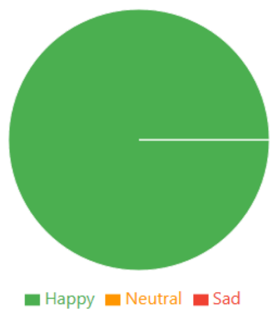


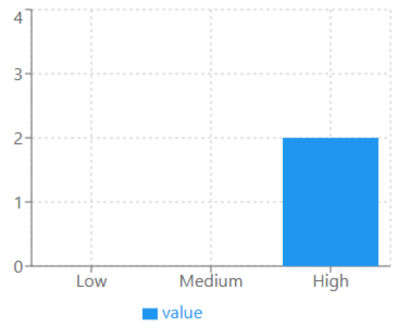
localhost:5173/dashboard

All Bookmarks

Date	Task Title	Time Spent	Mood	Energy Level	Notes	Actions
03/08/2025	Prepare Breakfast	60 min	happy	high	Prepared breakfast for family	Edit Delete

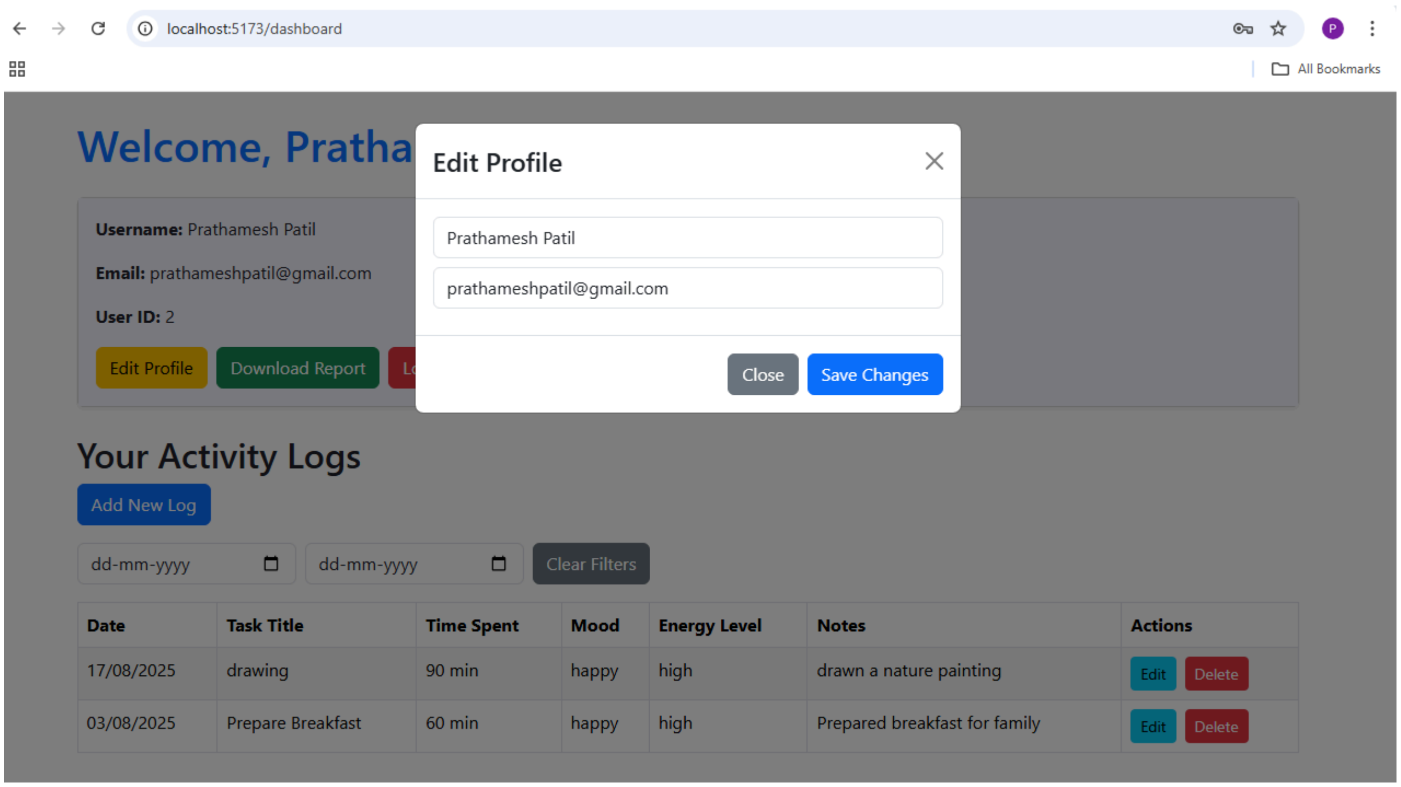
Your Analytics





💡 Suggestion

Keep it up 🎉 Great time for focused work!


localhost:5173/dashboard

Welcome, Pratha

Username: Prathamesh Patil

Email: prathameshpatil@gmail.com

User ID: 2

Edit Profile | Download Report | Lo

Edit Profile

Prathamesh Patil

prathameshpatil@gmail.com

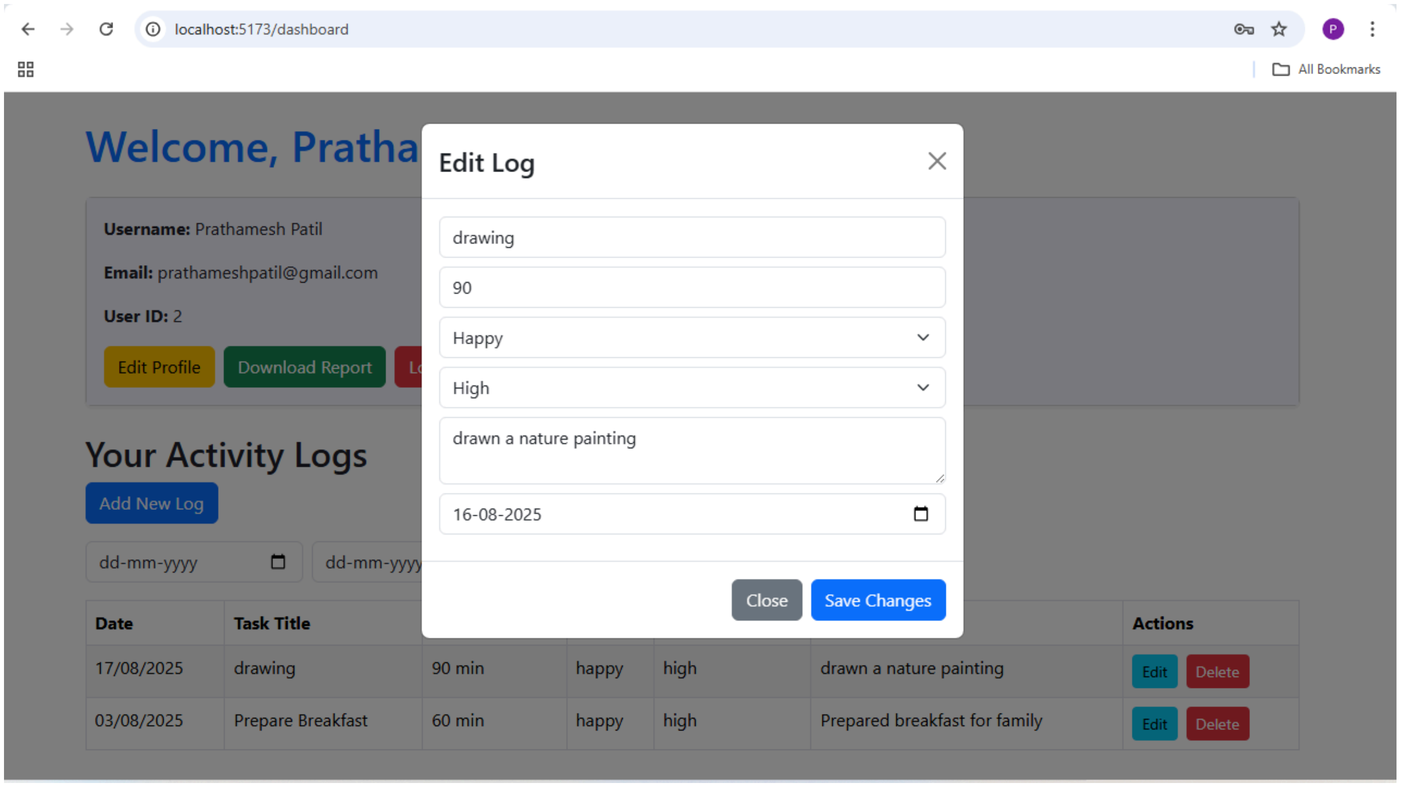
Close | Save Changes

Your Activity Logs

Add New Log

dd-mm-yyyy | dd-mm-yyyy | Clear Filters

Date	Task Title	Time Spent	Mood	Energy Level	Notes	Actions
17/08/2025	drawing	90 min	happy	high	drawn a nature painting	Edit Delete
03/08/2025	Prepare Breakfast	60 min	happy	high	Prepared breakfast for family	Edit Delete


localhost:5173/dashboard

Welcome, Pratha

Username: Prathamesh Patil

Email: prathameshpatil@gmail.com

User ID: 2

Edit Profile | Download Report | Lo

Edit Log

drawing

90

Happy

High

drawn a nature painting

16-08-2025

Close | Save Changes

Your Activity Logs

Add New Log

dd-mm-yyyy | dd-mm-yyyy

Date	Task Title					Actions
17/08/2025	drawing	90 min	happy	high	drawn a nature painting	Edit Delete
03/08/2025	Prepare Breakfast	60 min	happy	high	Prepared breakfast for family	Edit Delete

Coding

Logs Controllers

```
const pool = require('../config/db');

// Create a new log
const createLog = async (req, res) => {
  try {
    console.log('Inside createLog');

    const { task_title, time_spent, mood, energy_level, notes, date } = req.body;
    const user_id = req.user?.id; // Ensure auth middleware sets this

    if (!task_title || !time_spent || !date) {
      return res.status(400).json({ message: "Required fields are missing" });
    }

    const sql = `
      INSERT INTO logs (user_id, task_title, time_spent, mood, energy_level, notes, date)
      VALUES (?, ?, ?, ?, ?, ?, ?)
    `;

    const [result] = await pool.query(sql, [
      user_id,
      task_title,
      time_spent,
      mood || null,
      energy_level || null,
      notes || null,
      date,
    ]);

    res.status(201).json({
      message: "Log created successfully",
      logId: result.insertId,
    });

  } catch (error) {
    console.error("Error in createLog:", error);
    res.status(500).json({ message: "Server Error", error: error.message });
  }
};

// Get all logs (Admin use)
const getAllLogs = async (req, res) => {
  try {
    const [rows] = await pool.query('SELECT * FROM logs ORDER BY date DESC');

    res.status(200).json({
      message: 'Logs fetched successfully',
      result: rows
    });

  } catch (err) {
    console.error('Error fetching logs:', err);
    res.status(500).json({ message: 'Error fetching logs', error: err.message });
  }
};

// Get logs for a specific user
const getUserLogs = async (req, res) => {
  try {
    const { id } = req.params;

    const sql = 'SELECT * FROM logs WHERE user_id = ? ORDER BY date DESC';
    const [rows] = await pool.query(sql, [id]);

    if (rows.length === 0) {
      return res.status(200).json({ message: 'No logs found for this user', result: [] });
    }

    res.status(200).json({ message: 'Logs fetched successfully', result: rows });

  } catch (err) {
    console.error('Error in getUserLogs:', err);
    res.status(500).json({ message: 'Error fetching logs', error: err.message });
  }
};

// Update a log
const updateLog = async (req, res) => {
  try {
    const logId = req.params.id;
    const user_id = req.user?.id;
    const { task_title, time_spent, mood, energy_level, notes, date } = req.body;

    const sql = `
      UPDATE logs
      SET task_title = ?, time_spent = ?, mood = ?, energy_level = ?, notes = ?, date = ?
      WHERE id = ? AND user_id = ?
    `;
    const [result] = await pool.query(sql, [
      task_title,
      time_spent,
      mood || null,
      energy_level || null,
      notes || null,
      date,
      logId,
      user_id,
    ]);

    if (result.affectedRows === 0) {
      return res.status(404).json({ message: "Log not found or not authorized" });
    }

    res.status(200).json({ message: "Log updated successfully" });

  } catch (error) {
    console.error("Error in updateLog:", error);
    res.status(500).json({ message: "Server error", error: error.message });
  }
};

// Delete a log
const deleteLog = async (req, res) => {
  try {
    const { id } = req.params;
    const user_id = req.user?.id;

    const sql = "DELETE FROM logs WHERE id = ? AND user_id = ?";
    const [result] = await pool.query(sql, [id, user_id]);
    if (result.affectedRows === 0) {
      return res.status(404).json({ message: "Log not found or not authorized" });
    }

    res.status(200).json({ message: 'Log deleted successfully' });

  } catch (err) {
    console.error("Error deleting log:", err);
    res.status(500).json({ message: 'Error deleting log', error: err.message });
  }
};

module.exports = { createLog, getAllLogs, getUserLogs, updateLog, deleteLog};
```

Testing

Unit Testing: Unit testing where individual program units or object classes are tested. Here by using this testing we have focused on testing the functionality of methods.

Module Testing: Where this is the combination of unit program is called module. Here we tested the unit program (5-6 programs) is where the module programs have dependency.

Sub-system Testing: Then we combined some module for the Preliminary System Testing in our Project.

System Testing: Where it is the combination of two or more sub-system and then it is tested. Here we tested the Entire system as per the requirements.

Acceptance Testing: Normally this type of testing is done to verify if system meets the customer specified requirements. After submitting this project to User then they tested it and to determine whether to accept application. It is the system testing performed by the customer(s) to determine whether they should accept the delivery of the system.

6. Future scope

1. AI-driven Mood Prediction – Advanced machine learning models can be integrated to predict a user's mood based on their past logs and activity patterns.

2. Personalized Recommendations – More accurate lifestyle and productivity suggestions can be generated using data analytics and AI.

3. Mobile Application – A mobile version (Android/iOS) of the system can be developed to improve accessibility and usability.

4. Integration with Wearables – The system can be connected with smart devices (smartwatches, fitness trackers) to automatically capture physiological data such as heart rate, stress level, and sleep quality.

5. Data Sharing and Collaboration – Users can share their reports with healthcare professionals, mentors, or peers for better guidance and support.

6. Gamification Features – Features like streaks, achievements, and rewards can be introduced to keep users motivated and engaged.

7. Offline Mode with Syncing – An offline mode can be implemented where user data is stored locally and synchronized to the server once the internet connection is available.

Conclusion

The MoodMinder project successfully provides users with a simple yet effective way to track their daily moods, energy levels, and tasks. By combining data visualization with personalized suggestions, the system allows users to reflect on their emotional well-being and productivity.

The integration of analytics and PDF report generation makes the platform more valuable, as users can monitor progress over time and gain meaningful insights.

In conclusion, the project not only serves as a personal productivity and wellness tool, but also lays a strong foundation for future improvements. With the addition of advanced features such as AI-based predictions, mobile application support, and integration with wearables, MoodMinder can evolve into a comprehensive mental health and lifestyle management assistant.

BIBLIOGRAPHY

● *Web Development* by Ben Shaw & Saurabh

● *Development By Example* by Antonio Mele

REFERENCES

·

· MySQL Official Site – https://www.mysql.com/

· MySQL Tutorials – http://www.mysqltutorial.org

XAMPP Download & Documentation – https://www.apachefriends.org/download.html