<https://www.youtube.com/watch?v=CkVrmLLHmuI>

<https://laracasts.com/series/30-days-to-learn-laravel-11>

<https://laravel.com/>

<https://www.cloudways.com/blog/laravel-bootstrap-template-integration/>

<https://www.youtube.com/playlist?list=PL4cUxeGkcC9g9Vh9MAA-XKnfJsWZnPZFw>

<https://www.youtube.com/playlist?list=PLQDioScEMUhlIO3FLwmTsKCunfj6CK1Le>

<https://github.com/DoozieSoftware/quik_loans/invitations>

<https://webmail.dooziesoft.com/>

<https://laraveldaily.com/post/crud-how-to-avoid-building-whole-form-for-delete-button>

<https://www.youtube.com/watch?v=dKmjpDBxm2Y>

3. **\*\*Intro to Git & Version Control\*\***  
   - Learn Git branching strategy (feature branches, commits, pull requests).  
   - Create a **\*\*test branch\*\***, make a small change (fix a typo or add a console log), and push a commit.  
   - Raise a **\*\*dummy pull request (PR)\*\*** for review.

1. **\*\*Task: Create a “User Feedback” Feature\*\***  
   - Build a **\*\*basic form\*\*** (HTML/CSS + JavaScript) that takes user feedback (name, email, comments).  
   - Store the feedback in a **\*\*database table\*\***.  
   - Develop an **\*\*API endpoint\*\*** (`POST /feedback`) to save feedback into the database.  
   - Create an **\*\*API endpoint\*\*** (`GET /feedback`) to fetch all feedback data.2. **\*\*Additional Requirements:\*\***  
   - Use **\*\*React (if applicable) or simple HTML/JS\*\*** for the frontend.  
   - Backend to be implemented using **\*\*Node.js/PHP/Python (as per project stack)\*\***.  
   - Store feedback in a MySQL table.3. **\*\*Testing & Review\*\***  
   - Use **\*\*Postman\*\*** to test API requests and validate data storage.  
   - Create a simple **\*\*README file\*\*** explaining your approach.  
   - Raise a **\*\*pull request (PR)\*\*** for code review.---### **\*\*:pushpin: End-of-Week Checkpoint\*\***  
- **\*\*Demo session\*\***: Show your work and walk us through your approach.  
- **\*\*Feedback & improvements\*\***: Team will review your code and suggest improvements.  
- **\*\*Next steps\*\***: Based on progress, we will assign new tasks for the following week

<https://www.youtube.com/watch?v=W6NZfCO5SIk>

<https://www.youtube.com/watch?v=EUM78cxo0i8&list=PLDyQo7g0_nsVHmyZZpVJyFn5ojlboVEhE&index=3>

<https://github.com/Madhumohan08/doozie>

<https://github.com/Kavyanjali02/doozie/tree/kavya>

<https://github.com/Madhumohan08/userfeedback-form>

**Tasks for Today:**  
 :white_check_mark: Create a **new chatbot project** in [Dify.AI](http://dify.ai/).  
 :white_check_mark: Set up **basic Q&A intents** (e.g., "What is AI?", "What is Deep Learning?").  
 :white_check_mark: Upload relevant **engineering-related datasets** or documentation.  
 :white_check_mark: Fine-tune responses using prompt engineering.  
:pushpin: **Deliverables for Today:**  
 :one: A working chatbot with at least **10 AI/ML-related questions & answers**.  
 :two: A brief note on how you set up and trained it.

<https://dify101.com/>

<https://medium.com/@whiteking64/explore-dify-a-step-by-step-guide-to-building-a-simple-llm-powered-application-d1dd361a33ad>

<https://docs.dify.ai/>

* <https://dg.digantarashmi.com/login>, username : MSM
* password : MSM#123

<https://drive.google.com/drive/folders/1pk57AKIjV3heljyz0cxqT8d6Wb--XxeB?usp=sharing>

<https://www.youtube.com/watch?v=SWP3k-24jT4>

<https://youtu.be/sTeoEFzVNSc?si=kZeI4GXWhbkMKPUP>

<https://youtu.be/H4YK_7MAckk?si=A8RYLRF_BM01JlI7>

<https://youtu.be/_ZvnD73m40o?si=ky11PzkEgEXs2FhF>

<https://www.youtube.com/watch?v=XVO3zsHdvio>

<https://redmine.dooziesoft.co.in/>

**redmine.dooziesoft.co.in**

[**Doozie PM**](https://redmine.dooziesoft.co.in/)

Redmine

white_check_markeyesraised_hands

**NEW**

[3:11](https://dooziesoft.slack.com/archives/D08ELFRDB29/p1740390060523069)

akshay

PM software link redmine

[3:11](https://dooziesoft.slack.com/archives/D08ELFRDB29/p1740390068803779)

akshay

creds : username : madhumohan

[3:11](https://dooziesoft.slack.com/archives/D08ELFRDB29/p1740390075077359)

akshay

password : Temp#123

[3:11](https://dooziesoft.slack.com/archives/D08ELFRDB29/p1740390085215239)

akshay

let me know once you onboard

## :ear_of_rice: \*\*Maanay – Product Design Document (UI/UX Brief – Agriculture Focused)\*\*  
---### :bookmark: \*\*Product Name\*\*  
\*\*Maanay\*\* \*(“Daily Wage” in Konkani)\*  
> \*Empowering rural labor through simple, tech-enabled job access.\*---### :dart: \*\*Purpose\*\*  
Maanay is a mobile-first platform designed to connect \*\*agricultural daily wage workers\*\* with \*\*farm owners, cooperatives, and rural employers\*\*.It streamlines hiring for tasks like:  
- \*\*Field preparation\*\*  
- \*\*Fertilization & weeding\*\*  
- \*\*Climbing for areca, coconut, or pepper\*\*  
- \*\*Tractor driving & machine operation\*\*---### :busts_in_silhouette: \*\*Target Users\*\*#### 1. \*\*Agricultural Worker\*\*  
- Engaged in seasonal or recurring farm tasks  
- Often literate in local language only  
- Needs simple, icon-based, regional-language UI  
- Prefers cash or bank payments#### 2. \*\*Farmer / Plantation Owner\*\*  
- Needs flexible, verified labor for specific tasks  
- Often uses WhatsApp/YouTube but not complex apps  
- Needs trust, quick bookings, and clarity on worker skill---### :compass: \*\*Core Goals\*\*  
1. \*\*Task-specific matching\*\* (e.g., "banana cutting" or "weed spraying")  
2. \*\*Skill-based worker tagging\*\* for agriculture  
3. \*\*Availability calendar\*\* for seasonal/peak time  
4. \*\*Trust system:\*\* Ratings, prior job history, ID verified  
5. \*\*Simple booking + payment system\*\*---### :art: \*\*Design Direction\*\*  
- \*\*Tone:\*\* Rural-friendly, trustworthy, grounded  
- \*\*Colors:\*\* Earthy green, yellow, and soft brown  
- \*\*Fonts:\*\* Large fonts; localized (Kannada/Hindi/Marathi)  
- \*\*Icons:\*\* Agriculture-specific illustrations (tractor, sickle, sprayer, climber, etc.)  
- \*\*Language:\*\* Multilingual-first (localization is key)---### :frame_with_picture: \*\*Key Screens (Wireframe Scope – v1)\*\*#### :male-farmer: Worker Side  
1. \*\*Sign Up (Aadhaar + Mobile)\*\*  
   - Language selector, photo, basic profile  
2. \*\*Choose Skills (Icons)\*\*  
   - Tractor Driver, Weed Cutter, Climber, Fertilizer Spray, Banana Cutter  
3. \*\*Availability Schedule\*\*  
   - Day-wise toggle (Mon–Sun), hourly range  
4. \*\*Job Offer Screen\*\*  
   - Location, wage, crop type, time estimate  
5. \*\*Work Summary + Earnings\*\*  
   - Job list, daily/weekly payout, simple earnings chart---#### :female-farmer: Employer Side (Farmer)  
1. \*\*Sign Up (Mobile + Name)\*\*  
2. \*\*Post Job Requirement\*\*  
   - Skill needed → Date → No. of workers → Wages  
3. \*\*View Available Workers\*\*  
   - Filter by task (e.g. Sprayer or Climber), rating  
4. \*\*Confirm Booking\*\*  
   - ETA, wage, time → Confirm  
5. \*\*Payment & Review\*\*  
   - Cash or Online → Quick feedback form (emoji-based)---### :package: \*\*Deliverables for Designer\*\*  
- :white_check_mark: 1 \*\*User Flow Diagram\*\* (PDF or Figma)  
- :white_check_mark: 10 \*\*Wireframes\*\* (5 Worker + 5 Farmer side)

<https://chat.whatsapp.com/HZ9dUGVnd9OELTBx8QrTWM>

<https://youtu.be/R6ooI2_OaAk?si=isTbuZZQS1I0lCgc>

<https://quickstarts.postman.com/guide/php-laravel-API/index.html?index=..%2F..index#0>

**Project Overview**

* **Frontend (React.js + Tailwind + Recharts)** → Upload CSV & Display Graphs :bar_chart:
* **Backend (Flask + TextBlob + Pandas)** → Analyze Sentiments & Return Data :mag:
* **No Database Required** → Everything happens in memory :zap:

**:rocket: Step-by-Step Implementation**  
**:one: Backend (Flask)**  
Install dependencies:

bash

pip install flask flask-cors pandas textblob

C**reate**  
**app.py**  
This will: :white_check_mark: Accept CSV file uploads  
 :white_check_mark: Perform Sentiment Analysis  
 :white_check_mark: Return JSON data for the frontend

python

from flask import Flask, request, jsonify

import pandas as pd

from textblob import TextBlob

from flask\_cors import CORS

app = Flask(\_\_name\_\_)

CORS(app)

def analyze\_sentiment(comment):

analysis = TextBlob(comment)

polarity = analysis.sentiment.polarity

return "Positive" if polarity > 0 else "Negative" if polarity < 0 else "Neutral"

@app.route("/upload", methods=["POST"])

def upload\_file():

if "file" not in request.files:

return jsonify({"error": "No file uploaded"}), 400

file = request.files["file"]

df = pd.read\_csv(file, delimiter=";")

if "comments" not in df.columns:

return jsonify({"error": "CSV must contain 'comments' column"}), 400

df["Sentiment"] = df["comments"].apply(analyze\_sentiment)

sentiment\_counts = df["Sentiment"].value\_counts().to\_dict()

return jsonify({"feedback":

df.to\_dict(orient="records"), "stats": sentiment\_counts})

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**:two: Frontend (React.js)**  
**Set up React App**

bash

npx create-react-app sentiment-app

cd sentiment-app

npm install axios recharts tailwindcss

C**onfigure Tailwind**

bash

npx tailwindcss init

Modify  
t**ailwind.config.js**  
:

js

module.exports = {

content: ["./src/\*\*/\*.{js,jsx,ts,tsx}"],

theme: { extend: {} },

plugins: [],

};

Up**date**  
s**rc/index.css**

css

@tailwind base;

@tailwind components;

@tailwind utilities;

3️**⃣ React Frontend (**  
s**rc/App.js**  
**)**  
This will:  
 :white_check_mark: Allow users to upload a CSV file  
 :white_check_mark: Send file to Flask backend  
 :white_check_mark: Display sentiment analysis results  
 :white_check_mark: Show Pie Chart with Recharts

jsx

import React, { useState } from "react";

import axios from "axios";

import { PieChart, Pie, Tooltip, Cell } from "recharts";

const COLORS = ["#00C49F", "#FFBB28", "#FF8042"];

function App() {

const [file, setFile] = useState(null);

const [data, setData] = useState(null);

const handleFileChange = (e) => setFile(e.target.files[0]);

const handleUpload = async () => {

if (!file) return alert("Please upload a CSV file");

const formData = new FormData();

formData.append("file", file);

try {

const response = await axios.post("

<http://127.0.0.1:5000/upload>", formData, {

headers: { "Content-Type": "multipart/form-data" },

});

setData(response.data);

} catch (error) {

alert("Error uploading file!");

}

};

const chartData = data

? Object.keys(data.stats).map((key, index) => ({

name: key,

value: data.stats[key],

color: COLORS[index % COLORS.length],

}))

: [];

return (

<div className="min-h-screen flex flex-col items-center p-8 bg-gray-100">

<h1 className="text-2xl font-bold mb-4">Sentiment Analysis</h1>

<input type="file" onChange={handleFileChange} className="mb-4 p-2 border rounded" />

<button onClick={handleUpload} className="bg-blue-500 text-white px-4 py-2 rounded">

Upload & Analyze

</button>

{data && (

<div className="mt-6">

<h2 className="text-lg font-semibold">Results:</h2>

<PieChart width={400} height={400}>

<Pie data={chartData} cx={200} cy={200} outerRadius={120} dataKey="value">

{chartData.map((entry, index) => (

<Cell key={`cell-${index}`} fill={entry.color} />

))}

</Pie>

<Tooltip />

</PieChart>

</div>

)}

</div>

);

}

export default App;

**:four: Run the App**  
**Start Flask Backend**

bash

python app.py

S**tart React Frontend**

bash

npm start

**tep 1: Install Required Software**  
Before starting, ensure you have the necessary software installed:  
**1.1 Install XAMPP (Apache, MySQL, PHP)**

1. Download **XAMPP** from Apache Friends.
2. Install it and open the **XAMPP Control Panel**.
3. Start **Apache** (for PHP) and **MySQL** (for the database).

**1.2 Verify PHP and MySQL**

* Open your browser and go to:
* arduino

[http://localhost](http://localhost/)

* If XAMPP is running correctly, you'll see the XAMPP welcome page.

**Step 2: Set Up the MySQL Database**  
**2.1 Open phpMyAdmin**

1. Open **XAMPP Control Panel**.
2. Click on **Admin** next to MySQL, which will open **phpMyAdmin** in your browser.

**2.2 Create a Database**

1. Click **Databases** → Enter **feedback\_db** → Click **Create**.

**2.3 Create a Table**

1. Click **feedback\_db** → Click **SQL** and paste the following:
2. sql
3. CREATE TABLE feedback (
4. id INT AUTO\_INCREMENT PRIMARY KEY,
5. name VARCHAR(100) NOT NULL,
6. email VARCHAR(100) NOT NULL,
7. comments TEXT NOT NULL,
8. created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
9. );
10. Click
11. **Go** to execute the query.

**Step 3: Set Up the Project Folder**  
**3.1 Create Project Directory**

1. Navigate to C:\xampp\htdocs\ (or the htdocs folder in your XAMPP installation).
2. Create a new folder called **feedback\_project**.

**3.2 Create Necessary Files**  
Inside **feedback\_project**, create these files:

pgsql

feedback\_project/

│── index.html (Frontend Form)

│── db.php (Database Connection)

│── submit\_feedback.php (Handles form submission)

│── api.php (API Endpoints)

S**tep 4: Create the Frontend Feedback Form4.1 Create index.html**  
This is a simple HTML form for users to submit feedback.

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>User Feedback</title>

<style>

body { font-family: Arial, sans-serif; margin: 20px; padding: 20px; }

form { max-width: 400px; padding: 20px; border: 1px solid #ccc; border-radius: 5px; }

input, textarea { width: 100%; padding: 8px; margin: 5px 0; border: 1px solid #ccc; border-radius: 4px; }

button { padding: 10px; background-color: blue; color: white; border: none; cursor: pointer; }

</style>

</head>

<body>

<h2>Submit Your Feedback</h2>

<form id="feedbackForm">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="comments">Comments:</label>

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

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

</form>

<script>

document.getElementById("feedbackForm").addEventListener("submit", function(event) {

event.preventDefault();

let formData = new FormData(this);

fetch("submit\_feedback.php", {

method: "POST",

body: formData

})

.then(response => response.json())

.then(data => alert(data.message))

.catch(error => console.error("Error:", error));

});

</script>

</body>

</html>

S**tep 5: Backend Setup5.1 Create db.php (Database Connection)**

php

<?php

$servername = "localhost";

$username = "root"; // Change if needed

$password = ""; // Change if needed

$database = "feedback\_db";

// Create connection

$conn = new mysqli($servername, $username, $password, $database);

// Check connection

if ($conn->connect\_error) {

die(json\_encode(["error" => "Connection failed: " . $conn->connect\_error]));

}

?>

5**.2 Create**  
**submit\_feedback.php (Handling Form Submission)**

php

<?php

header("Content-Type: application/json");

include 'db.php';

if ($\_SERVER["REQUEST\_METHOD"] === "POST") {

$name = $\_POST["name"];

$email = $\_POST["email"];

$comments = $\_POST["comments"];

if (!empty($name) && !empty($email) && !empty($comments)) {

$stmt = $conn->prepare("INSERT INTO feedback (name, email, comments) VALUES (?, ?, ?)");

$stmt->bind\_param("sss", $name, $email, $comments);

if ($stmt->execute()) {

echo json\_encode(["message" => "Feedback submitted successfully!"]);

} else {

echo json\_encode(["error" => "Error submitting feedback"]);

}

$stmt->close();

} else {

echo json\_encode(["error" => "All fields are required"]);

}

} else {

echo json\_encode(["error" => "Invalid request method"]);

}

$conn->close();

?>

S**tep 6: Create API Endpoints6.1 Create api.php**  
This file will handle GET and POST requests for feedback.

php

<?php

header("Content-Type: application/json");

include 'db.php';

$request\_method = $\_SERVER["REQUEST\_METHOD"];

if ($request\_method == "POST") {

$data = json\_decode(file\_get\_contents("

<php://input>"), true);

if (isset($data["name"]) && isset($data["email"]) && isset($data["comments"])) {

$stmt = $conn->prepare("INSERT INTO feedback (name, email, comments) VALUES (?, ?, ?)");

$stmt->bind\_param("sss", $data["name"], $data["email"], $data["comments"]);

if ($stmt->execute()) {

echo json\_encode(["message" => "Feedback submitted successfully"]);

} else {

echo json\_encode(["error" => "Failed to submit feedback"]);

}

$stmt->close();

} else {

echo json\_encode(["error" => "Invalid input data"]);

}

}

elseif ($request\_method == "GET") {

$sql = "SELECT id, name, email, comments, created\_at FROM feedback ORDER BY created\_at DESC";

$result = $conn->query($sql);

$feedbacks = [];

while ($row = $result->fetch\_assoc()) {

$feedbacks[] = $row;

}

echo json\_encode($feedbacks);

}

else {

echo json\_encode(["error" => "Invalid request method"]);

}

$conn->close();

?>

**Test index.html in Browser** (<http://localhost/feedback_project/index.html>)  
enter details check in php myadmin if database is storing.**Step 7: Testing with Postman**  
**7.1 Test POST /feedback**

1. Open **Postman**.
2. Select **POST** method.
3. Use URL: <http://localhost/feedback_project/api.php>
4. Go to **Body → raw → JSON** and enter:
5. json
6. {
7. "name": "John Doe",
8. "email": "
9. [john@example.com](mailto:john@example.com)",
10. "comments": "Great service!"
11. }
12. Click **Send**.

**7.2 Test GET /feedback**

1. Select **GET** method.
2. Use URL: <http://localhost/feedback_project/api.php>
3. Click **Send** to see all feedback.

<https://app.wordtune.com/>

<https://tools.dsshub.in/>

<https://it-tools.tech/>

<https://www.youtube.com/watch?v=1SNZRCVNizg>