Feedback Flow is a simple full-stack web application that allows users (students, testers, or developers) to submit feedback or bug reports for their projects.

The app includes:

• A frontend form for submitting feedback.
• A backend REST API built using Node.js + Express to handle data.
• A JSON file for local data storage (no external DB needed).
• APIs that can be easily tested through TestSprite's mCP Server for endpoint validation and automation testing.

Tech Stack

Component	Technology Used
Frontend	HTML, CSS, Fetch API (JavaScript)
Backend	Node.js, Express.js
Storage	feedbacks.json (local JSON file)
Testing	TestSprite mCP Server (for API testing and automation)
Environment	Localhost (http://localhost:3000)

```
feedback-collector/
├── backend/
│   ├── server.js
│   ├── routes/
│   │   └── feedbackRoutes.js
│   ├── controllers/
│   │   └── feedbackController.js
│   ├── data/
│   │   └── feedbacks.json
│   └── utils/
│       └── fileHandler.js
├── frontend/
│   ├── index.html
│   ├── style.css
│   └── script.js
├── package.json
└── README.md
```

Backend APIs

1. POST /submit

Description: Used to submit new feedback data.

Request Example:

```
POST  http://localhost:3000/submit
Content-Type: application/json

{
  "name": "Prince Kumar",
  "email": "prince@example.com",
  "message": "Loved the UI and the simplicity of this project!"
}
```

Validation:
• All three fields (name, email, message) are required.
• Email should contain @.

Response Example:

```
{
  "status": "success",
  "message": "Feedback submitted successfully!"
}
```

2. GET /feedbacks

Description: Fetch all stored feedback entries.

Request Example:

```
GET http://localhost:3000/feedbacks
```

Response Example:

```
[
  {
    "id": 1,
    "name": "Prince Kumar",
    "email": "prince@example.com",
    "message": "Loved the UI and the simplicity of this project!",
    "timestamp": "2025-10-14T10:20:30.000Z"
  },
  {
    "id": 2,
{
  "name": "Aarav Singh",
  "email": "aarav@test.com",
  "message": "The backend API works perfectly!",
  "timestamp": "2025-10-14T10:25:42.000Z"
}
]
```

Frontend (index.html + script.js)

The frontend will include:

• A simple feedback form (name, email, message).
• A “Submit” button that sends data using fetch() to the /submit endpoint.
• A section that displays a success or error message after submission.

Example flow:

1. User fills the form →
2. Clicks “Submit Feedback” →
3. JS sends a POST request to /submit →
4. Backend validates and stores →
5. Confirmation message shown on the page.

Example Use Case (for TestSprite)

TestSprite can automate the following test cases:

Test Case	Endpoint	Method	Expected Result
1	/submit	POST	Returns success on valid data
2	/submit	POST	Returns error when fields are empty
3	/feedbacks	GET	Returns array of all feedbacks
4	/feedbacks	GET	Status 200 & JSON format
5	/submit	POST	Should append data to feedbacks.json

Project URLs

Type	URL
Frontend (Form Page)	http://localhost:3000
API: Submit Feedback	http://localhost:3000/submit
API: Get Feedbacks	http://localhost:3000/feedbacks

Example Output (Console/Terminal)

```
Server running on http://localhost:3000
New feedback received from Prince Kumar
Feedback stored successfully in data/feedbacks.json
```

Sample feedbacks.json (after few submissions)

```
[
  {
    "id": 1,
    "name": "Prince Kumar",
    "email": "prince@example.com",
    "message": "Really helpful tutorial!",
    "timestamp": "2025-10-14T10:20:30.000Z"
  },
  {
    "id": 2,
    "name": "Anjali Sharma",
    "email": "anjali@demo.com",
    "message": "This will help students learn basic APIs.",
    "timestamp": "2025-10-14T10:25:42.000Z"
```

🧠 Bonus: Testing Notes for TestSprite

• Ensure server runs on port 3000.
• Use JSON format for POST requests.
• Validate API response codes:
  ○ 200 → Success
  ○ 400 → Invalid Input

- Verify that feedbacks.json updates after valid submission.
- Use GET /feedbacks to cross-check stored feedback count.