### **Simple Task Manager API — 15-Day Plan**

**Team Members:**

* Abhishek
* Harsh

### **Project Overview**

Build a simple RESTful API using Node.js and Express.js to manage tasks.  
 Features:

* Create, read, update, delete tasks (CRUD)
* Mark tasks as complete
* Filter tasks by status
* Use JSON file storage or in-memory array

### **Tools & Tech Stack**

* Node.js
* Express.js
* Postman (for testing)
* Optional: JSON file for data persistence
* Deployment platforms: Render or Railway

### **Folder Structure (Recommended)**

pgsql

CopyEdit

simple-task-manager/

├── controllers/

│ └── taskController.js

├── routes/

│ └── taskRoutes.js

├── utils/

│ └── fileStorage.js (optional)

├── data/

│ └── tasks.json

├── middleware/

│ └── logger.js

├── server.js

└── package.json

### **Daily Task Breakdown (2 hours/day)**

| **Day** | **Task Description** | **Abhishe's Responsibility** | **Harsh's Responsibility** |
| --- | --- | --- | --- |
| **1** | Setup Node project | Install Node.js, create project folder, npm init | Learn npm basics, create test script |
| **2** | Setup Express server | Install Express, create server.js, set up basic server | Create base route / returning "Hello World", test route |
| **3** | Define Task model | Create task object structure (id, title, description, completed) | Setup tasks.js file with tasks array export |
| **4** | Create task API (POST /tasks) | Write POST route, add new task to array | Add validation (title required), test with Postman |
| **5** | List all tasks (GET /tasks) | Implement GET route to return all tasks | Format response properly, handle empty list |
| **6** | Get task by ID (GET /tasks/:id) | Fetch task by ID from array | Handle "task not found" errors |
| **7** | Update task (PUT /tasks/:id) | Implement updating task info (title, description) | Add validation, test via Postman |
| **8** | Delete task (DELETE /tasks/:id) | Implement deletion by ID | Return success/failure messages |
| **9** | Mark task complete (PATCH /tasks/:id/complete) | Update task completed status to true | Handle edge cases, test route |
| **10** | API Testing | Test all routes via Postman for correctness | Document endpoints, create API reference |
| **11** | Middleware | Create logging middleware to log request details | Apply middleware to all routes |
| **12** | File-based storage (optional) | Write utility to save tasks array to tasks.json | Implement reading from tasks.json on startup |
| **13** | Error Handling | Create custom error responses | Handle input validation errors gracefully |
| **14** | Code Refactor | Move routes to routes/taskRoutes.js | Move business logic to controllers/taskController.js |
| **15** | Deployment | Deploy app on Render or Railway | Test live app, share deployment link |

### **Bonus Features (optional)**

* Add user authentication with JWT
* Add task due dates and sorting
* Add unit tests with Jest

### **Useful Resources**

* Node.js Official Docs
* Express.js Guide
* [Postman for API Testing](https://www.postman.com/)
* Render Deployment
* Railway Deployment

### **Notes for Abhishe & Harsh**

* Communicate daily to sync up progress.
* Push code regularly to a shared GitHub repo.
* Use Git branches for each feature or day.
* Review each other's code for learning and quality.