

TaskMasterAPI – Server-Side Web Application Testing Project

TaskMasterAPI is a Node.js + Express backend designed for **server-side web application modeling and testing**.

The project demonstrates:

- ✓ Construction of a **Conceptual Interaction Model (CIM)**
 - ✓ Construction of an **Abstract Test Graph (ATG)**
 - ✓ Mapping of tests to ATG edges for **coverage-driven analysis**
 - ✓ Implementation of **REST APIs** for Users, Sessions (Login), and Tasks
 - ✓ Automated testing using **Mocha + Chai + Supertest**
 - ✓ Achieving **100% ATG edge coverage**
-

1. Project Overview

TaskMasterAPI is a lightweight task-management backend with three major modules:

1. User Management

- Create new user
- Prevent duplicate emails

2. Authentication (Sessions)

- JWT-based login
- Session stored in DB for verification

3. Tasks Module

- Create, update, delete tasks
- List tasks belonging to a specific user
- Enforces strict authorization

The system is intentionally designed to contain multiple **decision points**, which are represented in the **ATG (Abstract Test Graph)** and used for deriving a comprehensive test suite.

2. Features

Users

- POST `/users`
- Validates missing fields
- Prevents duplicate email
- Returns user ID upon creation

Authentication

- POST `/sessions/login`
- Issues JWT token
- Handles invalid credentials

Tasks

All task routes require authentication via JWT.

- POST `/tasks` – create task
- GET `/tasks/users/:id/tasks` – list tasks for a user
- PUT `/tasks/:id` – update task
- DELETE `/tasks/:id` – delete task

Middleware

- Authorization middleware validates tokens
- Ensures only resource owners can access/modify tasks

3. Project Structure

```
TaskMasterAPI/
|
└── src/
    ├── app.js
    ├── config/db.js
    └── controllers/
        └── userController.js
```

```
|- |
|   |- authController.js
|   |- taskController.js
|- middleware/
|   |- authMiddleware.js
|- models/
|   |- user.js
|   |- task.js
|   |- session.js
|- routes/
|   |- userRoutes.js
|   |- authRoutes.js
|   |- taskRoutes.js
|
|- tests/
|   |- mocha_tests.js
|   |- mocha_additional_tests.js
|
|- ATG.pdf
|- CIM.pdf
|- ATG for TaskMaster.drawio.png
|
|- README.md
```

4. Conceptual Interaction Model (CIM)

The **CIM** describes the high-level interaction between:

- User → API → Controllers → DB
- Login flow
- Authenticated task operations
- Error and alternative flows

This model is included as:

- **CIM.pdf**
- **ATG for TaskMaster.drawio.png (visual diagram)**

It forms the conceptual foundation for the **ATG**.

5. Abstract Test Graph (ATG)

The **Abstract Test Graph (ATG)** captures:

- All server-side decisions
- All success and failure branches
- Alternative flows (missing fields, bad credentials, forbidden access, DB errors, etc.)
- Relations between request → middleware → controller → DB outcome

The ATG is provided in:

- **ATG.pdf**
- **ATG for TaskMaster.drawio.png**

The graph has **23 distinct edges**, each representing a unique behavior or branch in the system.

6. Model-Based Test Design

Two test suites are provided:

mocha_tests.js

Covers all **basic flows**:

- Create user
- Login
- Create task
- List tasks

mocha_additional_tests.js

Covers all **negative, boundary, and error flows**:

- Missing fields
 - Duplicate email
 - Invalid login
 - Missing/invalid token
 - Forbidden access
 - Missing title
 - Non-existent task update
 - Successful delete + delete-not-found
 - Unauthorized update attempts
-

7. Test-to-ATG Mapping (100% Coverage)

| Test ID | Test Name | ATG Edges |
|---------|---------------------------|----------------|
| T1 | Basic create user | 1, 2 |
| T2 | Login | 5, 6, 8 |
| T3 | Create task | 10, 11 |
| T4 | List tasks | 13, 14 |
| T5 | Missing user fields | 1, 3 |
| T6 | Duplicate email | 1, 4 |
| T7 | Invalid login | 5, 7 |
| T8 | Missing or invalid token | 10, 9 |
| T9 | Create task missing title | 10, 12 |
| T10 | Forbidden listing | 13, 15 |
| T11 | Update non-existent | 16, 18 |
| T12 | Create + delete | 10, 11, 20, 21 |
| — | Delete not found | 20, 22 |
| — | Unauthorized update | 16, 19 |

✓ All 23 edges in ATG covered

✓ Final coverage = 100%

This satisfies academic requirements for **full model-based testing** coverage.

8. How to Run the Project

Install dependencies

```
npm install
```

Configure environment

Create a `.env` file:

```
MONGO_URI=mongodb://localhost:27017/taskmaster  
PORT=3000  
JWT_SECRET=your_secret
```

Run server

```
npm start
```

Run tests

```
npm test
```

Mocha will run both test suites and automatically clean the database before & after execution.

9. Technologies Used

- **Node.js**
- **Express.js**
- **MongoDB + Mongoose**
- **Mocha** (test runner)
- **Chai** (assertions)
- **Supertest** (HTTP request testing)
- **JWT Authentication**