KOCAELI UNIVERSITY

Faculty of Engineering
Department of Software Engineering

Software Development I

Midterm Project Report

Task Management Web Application

Name & Surname:	Sena Nur Dülger	**Student ID:**	220229053
Name & Surname:	Irmak Yılmaz	**Student ID:**	230229038

Date: November 11, 2025

V 1.0

Task Management Web Application

Abstract—**This report presents the design and implementation of a web-based Task Management application built with the MERN stack (MongoDB, Express.js, React, Node.js). The system enables registered users to create, update, categorize, and track tasks by status and due date, while enforcing secure authentication and robust data handling. The report focuses only on the required evidence items.**

Index Terms—**task management, MERN stack, React, Node.js, Express.js, MongoDB, REST API, JSON Web Token, JWT, bcrypt, authentication, password hashing, data visualization, frontend backend integration, database connectivity, error handling, validation**

I. Introduction

Task management and productivity-focused web applications have become one of the fastest-evolving areas of full-stack development. Addressing the everyday needs of businesses and individuals—planning, team coordination, and tracking deadlines—this project delivers a secure and scalable Task Management application using the MERN architecture. The app follows contemporary practices: a React UI consumes a REST API built with Node.js/Express; data persist in MongoDB; authentication is token-based with JWT; and user passwords are stored as salted bcrypt hashes. This report covers the following sections:

1) *Introduction:* Summary of the topic and scope of the report.
2) *Technologies Used:* Chosen stack and brief rationale.
   a) Frontend (UI)
   b) Backend (API)
   c) Database
   d) Security and Encryption
   e) Additional Tools
3) *Implementation Results:* Evidence of a working integration.
   a) Frontend + Backend Integration
   b) Database Connection
4) *Interface Draft:* Initial appearance of the main screens.
5) *Encrypted Password:* Example bcrypt hash and brief explanation.
6) *Invalid Login Outcome:* 401 response and user-facing message.
7) *Conclusion:* Short evaluation and next steps.
8) *References*

II. Technologies Used

This section briefly summarizes the technologies used in the project and the reasons for their selection.

Technology	**Selection**	**Reasons**
Frontend (UI)	React + Vite	React was selected for its component-based architecture and vibrant ecosystem, enabling fast development and maintainable UIs. Combined with Vite and React Router, it provides instant hot-module refresh, efficient routing, and route-level code splitting for improved performance. Styling is handled using standard CSS modules for a clean and lightweight design.
Backend (API)	Node.js + Express	Node.js with Express was selected for its lightweight, event-driven runtime and mature middleware ecosystem. It makes building RESTful endpoints straightforward, adding JWT/httpOnly-cookie authentication, input validation, CORS, and centralized error handling, while integrating cleanly with a MongoDB data layer for a simple end-to-end JavaScript stack.
Database	MongoDB	MongoDB was selected for its flexible, JSON/BSON-based document model. It integrates seamlessly with the MERN stack and is ideal for storing variable data structures like user tasks.
Security and Encryption	Bcrypt + JWT	Bcrypt was chosen to securely hash user passwords. It incorporates salting to protect against rainbow table attacks and stores hashes irreversibly in the database. JWT was selected to create secure, stateless authentication tokens, allowing the server to verify API requests without storing session data.

TABLE I
Technology Selections and Reasons (Part I)

Technology	Selection	Reasons
Additional Tools	Chart.js, Axios, Nodemon, Jest, Supertest, Multer	Chart.js, Axios, and Nodemon were selected to streamline visualization, networking, and developer ergonomics. Chart.js delivers responsive charts with minimal setup; Axios provides concise HTTP calls with built-in auth and error handling; and Nodemon speeds backend iteration by auto-restarting on file changes, improving productivity and maintaining a smooth development workflow. Jest and Supertest were integrated to establish a robust backend testing environment, allowing for automated verification of API endpoints and ensuring system stability. Additionally, Multer was chosen as the middleware for handling multipart/form-data, providing the essential functionality for secure file uploads and storage management.

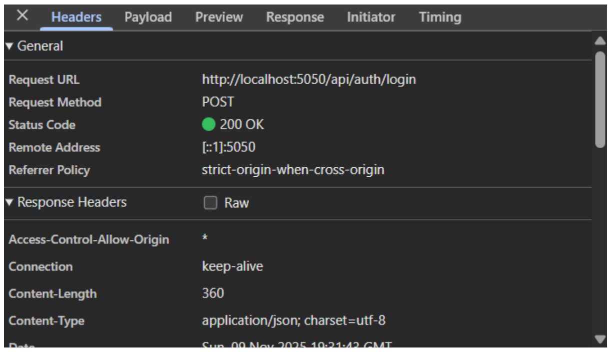
TABLE II
TECHNOLOGY SELECTIONS AND REASONS (PART II)

III. IMPLEMENTATION RESULTS

Evidence of a working integration, including frontend + backend integration and an active database connection.

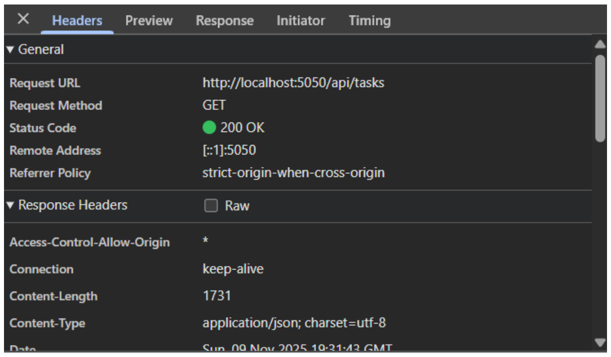
A. Frontend + Backend Integration

The first figure shows the browser's network tab capturing a successful POST request from the sign-in interface to `/api/users/login`, which returned 200 OK, confirming the connection and successful authentication. The second figure shows a subsequent GET request to `/api/tasks` returning 200 OK with JSON data.



```
▼ General
Request URL                  http://localhost:5050/api/auth/login
Request Method               POST
Status Code                  ● 200 OK
Remote Address               [::1]:5050
Referrer Policy              strict-origin-when-cross-origin
▼ Response Headers           ☐ Raw
Access-Control-Allow-Origin  *
Connection                   keep-alive
Content-Length               360
Content-Type                 application/json; charset=utf-8
Date                         Sun, 09 Nov 2025 19:31:43 GMT
```

Fig. 1. POST `/api/users/login` — 200 OK

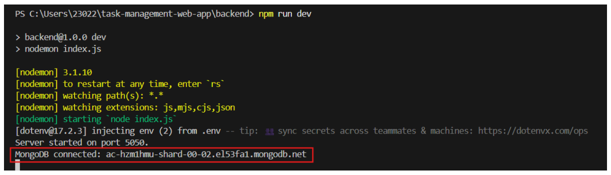


```
▼ General
Request URL                  http://localhost:5050/api/tasks
Request Method               GET
Status Code                  ● 200 OK
Remote Address               [::1]:5050
Referrer Policy              strict-origin-when-cross-origin
▼ Response Headers           ☐ Raw
Access-Control-Allow-Origin  *
Connection                   keep-alive
Content-Length               1731
Content-Type                 application/json; charset=utf-8
Date                         Sun, 09 Nov 2025 19:31:43 GMT
```

Fig. 2. GET `/api/tasks` — 200 OK

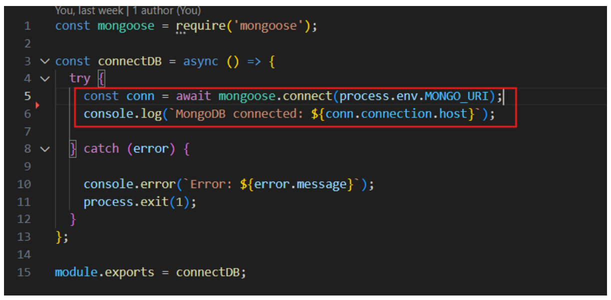
B. Database Connection

The backend establishes a persistent database session by initializing Mongoose with the environment variable `MONGO_URI`. On success, it logs the connected host (e.g., *MongoDB Connected: <host>*), confirming that queries will persist and read data from the live database during runtime.



```
PS C:\Users\23022\task-management-web-app\backend> npm run dev

> backend@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
[dotenv@17.2.3] injecting env (2) from .env -- tip: sync secrets across teammates & machines: https://dotenvx.com/ops
Server started on port 5050.
MongoDB connected: ac-hzm1hmu-shard-00-02.el53fa1.mongodb.net
```

Fig. 3. Terminal: successful connection log



```
const mongoose = require('mongoose');

const connectDB = async () => {
  try {
    const conn = await mongoose.connect(process.env.MONGO_URI);
    console.log(`MongoDB connected: ${conn.connection.host}`);

  } catch (error) {

    console.error(`Error: ${error.message}`);
    process.exit(1);
  }
};

module.exports = connectDB;
```

Fig. 4. Code: Mongoose connect using `MONGO_URI`

IV. INTERFACE DRAFT

This section presents the initial interface design of the *TaskMint* application.

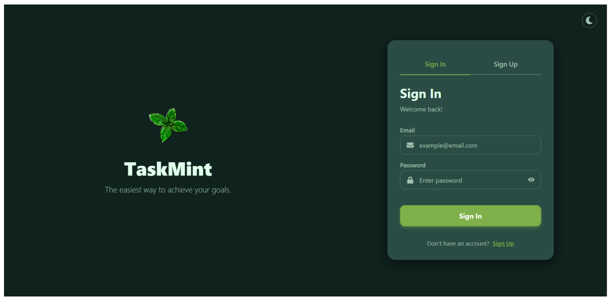


Fig. 5. Sign In

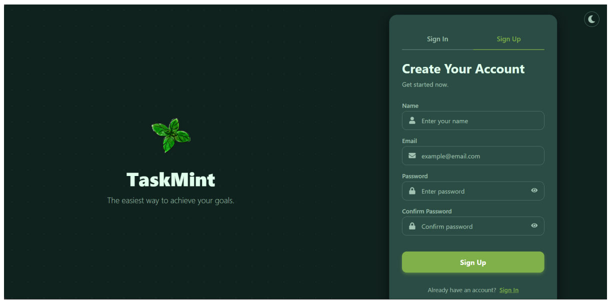
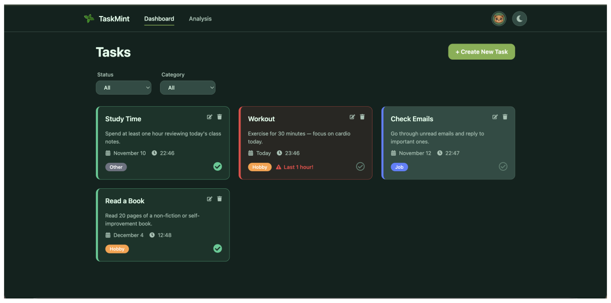


Fig. 6. Sign Up



Study Time
Spend at least one hour reviewing today's class notes.
November 10 22:46
Other

Workout
Exercise for 30 minutes — focus on cardio today.
Today 23:46
Hobby Last 1 hour!

Check Emails
Go through unread emails and reply to important ones.
November 12 22:47
Job

Read a Book
Read 20 pages of a non-fiction or self-improvement book.
December 4 12:48
Hobby

Fig. 7. Dashboard

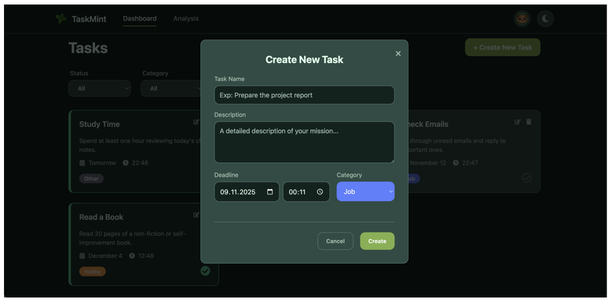


Fig. 8. Create New Task

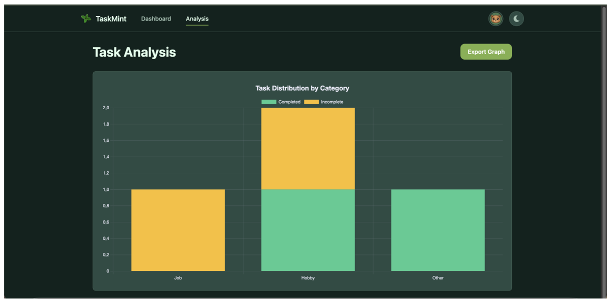


Fig. 9. Analysis

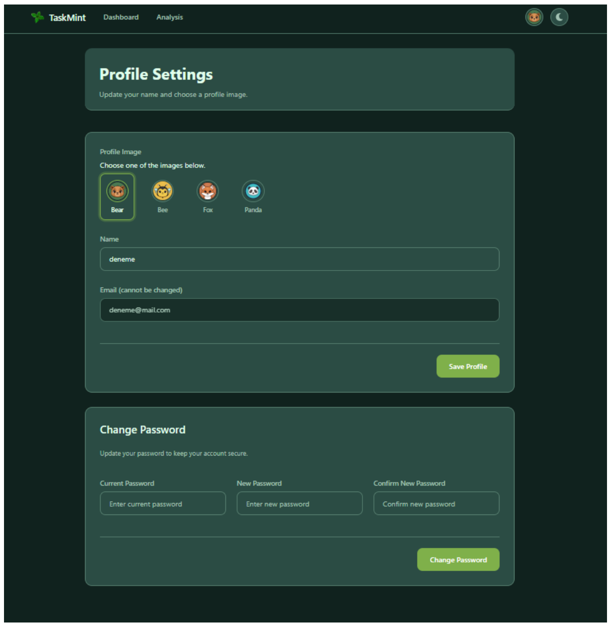
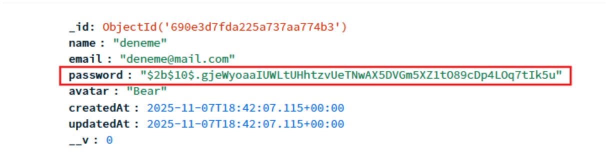


Fig. 10. Profile Settings

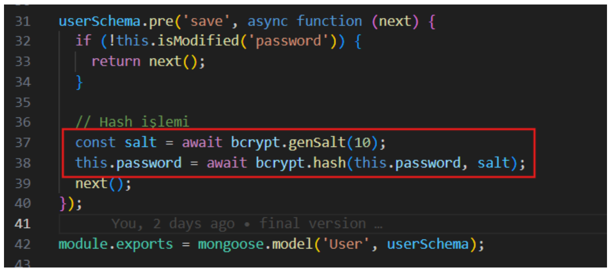
The design adopts a cohesive dark theme with mint accents. Figs. 5. - 6. show the entry flow with tabbed authentication, validated inputs, and a single primary action. Fig. 7. presents a card-based dashboard where each task exposes title, details, due date/time, status, and category chips with inline actions. Fig. 8. illustrates the creation modal that captures name, description, deadline (date & time) and category with clear cancel/confirm affordances. Fig. 9. summarizes the completion by category in a readable bar chart with export support, and Fig. 10. provides profile editing with avatar selection and a password-change panel. The Edit Task view reuses the same modal layout as Fig.8.; all fields are pre-filled with the selected task's data and the primary action becomes Save. The general spacing, hierarchy, and typography are tuned for clarity and responsive behavior.

V. Encrypted Password



```
_id: ObjectId('690e3d7fda225a737aa774b3')
name : "deneme"
email : "deneme@mail.com"
password : "$2b$10$.gjeWyoaaIUWLtUHhtzvUeTNwAX5DVGm5XZ1tO89cDp4LOq7tIk5u"
avatar : "Bear"
createdAt : 2025-11-07T18:42:07.115+00:00
updatedAt : 2025-11-07T18:42:07.115+00:00
__v : 0
```

Fig. 11. Hashed value in DB



```
userSchema.pre('save', async function (next) {
  if (!this.isModified('password')) {
    return next();
  }

  // Hash işlemi
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
  next();
});
      You, 2 days ago • final version …
module.exports = mongoose.model('User', userSchema);
```

Fig. 12. Hashing in code

Passwords are not stored in plain text. As shown in Fig. 11 and Fig. 12, the system generates a random salt using `bcrypt.genSalt(10)` and then hashes the password with `bcrypt.hash()`. Thanks to the salt and the cost factor (10 rounds), the same password produces different hashes for different users and is computationally hard to reverse. Only the salted hash is saved to the database, which prevents exposure of the original password even if the database is leaked.

VI. Invalid Login Outcome

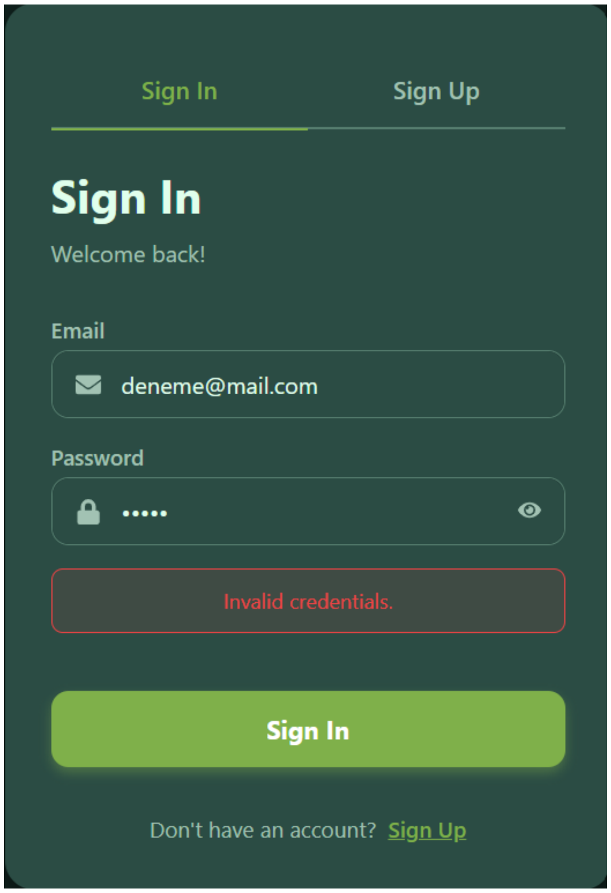
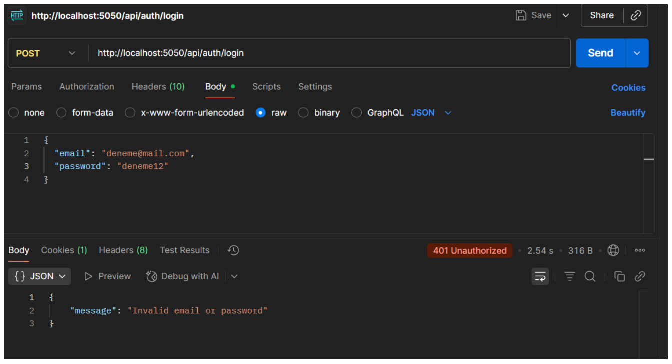
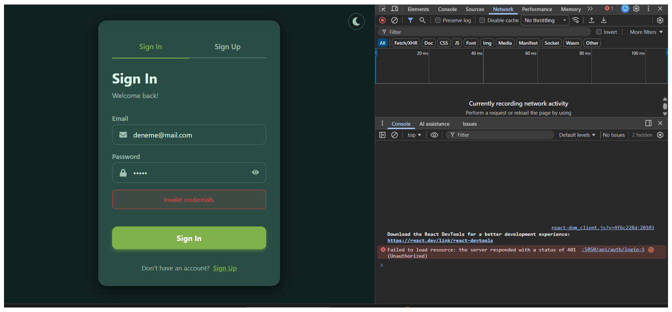


Fig. 13. UI: "Invalid credentials."



```
POST http://localhost:5050/api/auth/login
Body: raw JSON
{
  "email": "deneme@mail.com",
  "password": "deneme12"
}

Body  Cookies (1)  Headers (8)  Test Results    401 Unauthorized · 2.54 s · 316 B
{
    "message": "Invalid email or password"
}
```

Fig. 14. API client: 401 Unauthorized (Postman)



```
                                                   react-dom_client.js?v=4f6c228d:20103
Download the React DevTools for a better development experience:
https://react.dev/link/react-devtools
Failed to load resource: the server responded with a status of 401   :5050/api/auth/login:1
(Unauthorized)
```

Fig. 15. Combined view (UI + console)

When an incorrect email or password is submitted, the system rejects the request with an HTTP 401 Unauthorized response and a generic message. The UI surfaces "Invalid credentials.", no session or token is issued, and the flow cleanly handles both success and error paths without leaking sensitive details.

VII. Conclusion

All midterm requirements for the Task Management Web Application are complete. The core MERN stack is operational, including an active database connection, successful frontend-backend integration, and secure JWT/Bcrypt authentication. The initial UI draft is also finished. Future work will now focus on implementing full task (CRUD) operations, automated backend testing, and the final "Analysis" dashboard.

References

[1] "Build a Full-Stack MERN Task Manager — React, Node.js, MongoDB, Express — MERN Project," YouTube, [Online Video]. Accessed: Nov. 9, 2025. [Online]. Available: https://www.youtube.com/watch?v=fZK57PxKC-0&list=PLLSbWBaXAMvKZw4F1qwYPuDxwYF_gq7i4

[2] "Build a Full-Stack MERN Project Task Manager Tool — React.js + Node.js — MERN (2025)," YouTube, [Online Video]. Accessed: Nov. 9, 2025. [Online]. Available: https://www.youtube.com/watch?v=VAKDr1lsix0&list=PLLSbWBaXAMvKZw4F1qwYPuDxwYF_gq7i4&index=2

[3] "MERN Stack Project — Fullstack Tutorial," YouTube, [Online Video]. Accessed: Nov. 9, 2025. [Online]. Available: https://www.youtube.com/watch?v=H-9l-gTq-C4&list=PL0Zuz27SZ-6P4dQUsoDatjEGpmBpcOW8V

[4] "Connect frontend and backend — React JS, Node JS, Express — Send data from backend to frontend — API," YouTube, [Online Video]. Accessed: Nov. 9, 2025. [Online]. Available: https://www.youtube.com/watch?v=OgYKAI38JvY&list=PLLSbWBaXAMvKZw4F1qwYPuDxwYF_gq7i4&index=10

[5] "Node.js Full Course for Beginners — Complete All-in-One Tutorial — 7 Hours," YouTube, [Online Video]. Accessed: Nov. 9, 2025. [Online]. Available: https://www.youtube.com/watch?v=f2EqECiTBL8&list=PLLSbWBaXAMvKZw4F1qwYPuDxwYF_gq7i4&index=11

[6] "Understand MERN Stack," GeeksforGeeks, [Online]. Accessed: Nov. 9, 2025. [Online]. Available: https://www.geeksforgeeks.org/mern/understand-mern-stack/

[7] OpenAI, "ChatGPT," [Online]. Accessed: Nov. 9, 2025. [Online]. Available: https://openai.com/chatgpt

[8] Google, "Gemini," [Online]. Accessed: Nov. 9, 2025. [Online]. Available: https://gemini.google.com

[9] Google, "Stitch," [Online]. Accessed: Nov. 9, 2025. [Online]. Available: https://stitch.withgoogle.com/

KOCAELI UNIVERSITY

Faculty of Engineering
Department of Software Engineering

Software Development I

Final Project Report

Task Management Web Application

Name & Surname:	Sena Nur Dülger	**Student ID:**	220229053
Name & Surname:	Irmak Yılmaz	**Student ID:**	230229038

Date: December 24, 2025

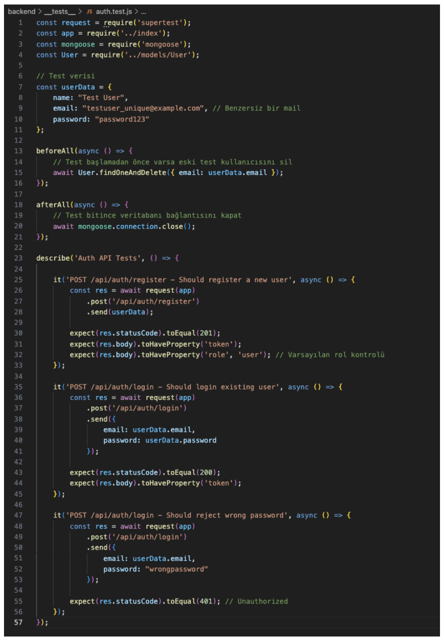
V 2.0

VIII. Backend Testing & Validation

To verify backend reliability, integration tests were implemented using Jest and Supertest. The suite sends real HTTP requests to the Express API and validates status codes, JWT-protected access, and MongoDB persistence.

A. *Test Scenarios (Code)*

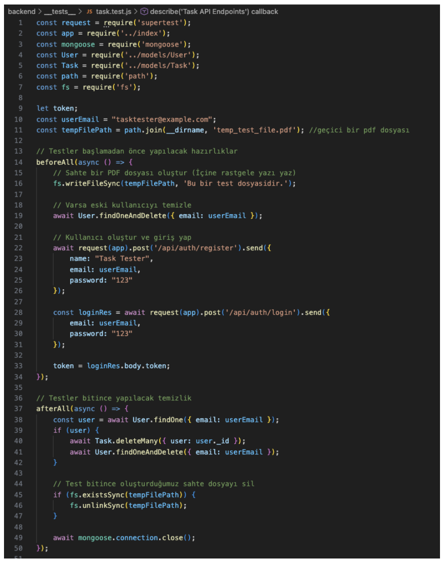
1) Authentication (Register/Login): These tests verify successful registration and login, and ensure invalid credentials are rejected with `401 Unauthorized`.



```
backend > __tests__ > JS auth.test.js > ...
const request = require('supertest');
const app = require('../index');
const mongoose = require('mongoose');
const User = require('../models/User');

// Test verisi
const userData = {
    name: "Test User",
    email: "testuser_unique@example.com", // Benzersiz bir mail
    password: "password123"
};

beforeAll(async () => {
    // Test başlamadan önce varsa eski test kullanıcısını sil
    await User.findOneAndDelete({ email: userData.email });
});

afterAll(async () => {
    // Test bitince veritabanı bağlantısını kapat
    await mongoose.connection.close();
});

describe('Auth API Tests', () => {

    it('POST /api/auth/register - Should register a new user', async () => {
        const res = await request(app)
            .post('/api/auth/register')
            .send(userData);

        expect(res.statusCode).toEqual(201);
        expect(res.body).toHaveProperty('token');
        expect(res.body).toHaveProperty('role', 'user'); // Varsayılan rol kontrolü
    });

    it('POST /api/auth/login - Should login existing user', async () => {
        const res = await request(app)
            .post('/api/auth/login')
            .send({
                email: userData.email,
                password: userData.password
            });

        expect(res.statusCode).toEqual(200);
        expect(res.body).toHaveProperty('token');
    });

    it('POST /api/auth/login - Should reject wrong password', async () => {
        const res = await request(app)
            .post('/api/auth/login')
            .send({
                email: userData.email,
                password: "wrongpassword"
            });

        expect(res.statusCode).toEqual(401); // Unauthorized
    });
});
```

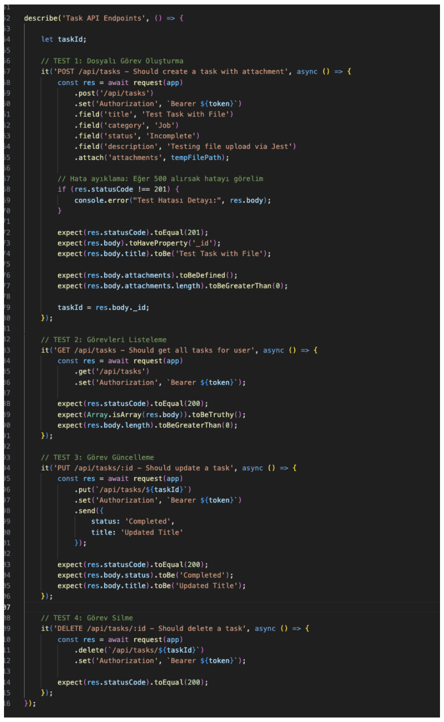
Fig. 16. Authentication test scenarios

2) Task API (CRUD + File Upload): Task endpoints are tested end-to-end, including protected routes with `Bearer` JWT, CRUD operations, and multipart file upload via `.field() + .attach()`.



```
backend > __tests__ > JS task.test.js > describe('Task API Endpoints') callback
const request = require('supertest');
const app = require('../index');
const mongoose = require('mongoose');
const User = require('../models/User');
const Task = require('../models/Task');
const path = require('path');
const fs = require('fs');

let token;
const userEmail = "tasktester@example.com";
const tempFilePath = path.join(__dirname, 'temp_test_file.pdf'); //geçici bir pdf dosyası

// Testler başlamadan önce yapılacak hazırlıklar
beforeAll(async () => {
    // Sahte bir PDF dosyası oluştur (İçine rastgele yazı yaz)
    fs.writeFileSync(tempFilePath, 'Bu bir test dosyasidir.');

    // Varsa eski kullanıcıyı temizle
    await User.findOneAndDelete({ email: userEmail });

    // Kullanıcı oluştur ve giriş yap
    await request(app).post('/api/auth/register').send({
        name: "Task Tester",
        email: userEmail,
        password: "123"
    });

    const loginRes = await request(app).post('/api/auth/login').send({
        email: userEmail,
        password: "123"
    });

    token = loginRes.body.token;
});

// Testler bitince yapılacak temizlik
afterAll(async () => {
    const user = await User.findOne({ email: userEmail });
    if (user) {
        await Task.deleteMany({ user: user._id });
        await User.findOneAndDelete({ email: userEmail });
    }

    // Test bitince oluşturduğumuz sahte dosyayı sil
    if (fs.existsSync(tempFilePath)) {
        fs.unlinkSync(tempFilePath);
    }

    await mongoose.connection.close();
});
```

Fig. 17. Test setup

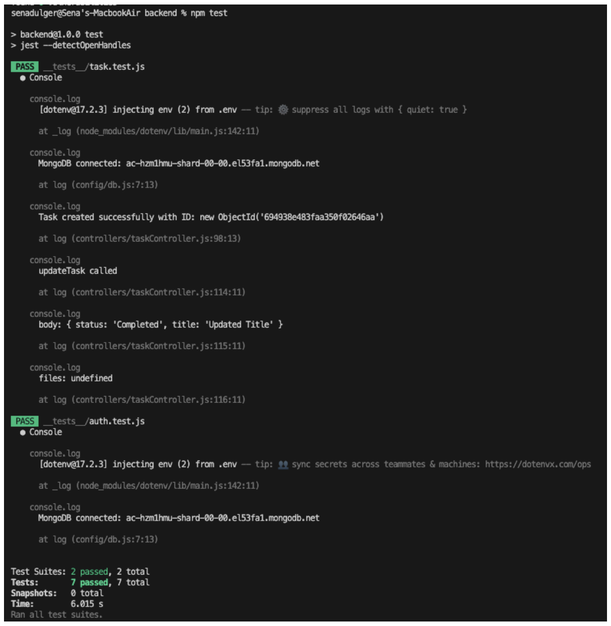


```
describe('Task API Endpoints', () => {

    let taskId;

    // TEST 1: Dosyalı Görev Oluşturma
    it('POST /api/tasks - Should create a task with attachment', async () => {
        const res = await request(app)
            .post('/api/tasks')
            .set('Authorization', `Bearer ${token}`)
            .field('title', 'Test Task with File')
            .field('category', 'Job')
            .field('status', 'Incomplete')
            .field('description', 'Testing file upload via Jest')
            .attach('attachments', tempFilePath);

        // Hata ayıklama: Eğer 500 alırsak hatayı görelim
        if (res.statusCode !== 201) {
            console.error("Test Hatası Detayı:", res.body);
        }

        expect(res.statusCode).toEqual(201);
        expect(res.body).toHaveProperty('_id');
        expect(res.body.title).toBe('Test Task with File');

        expect(res.body.attachments).toBeDefined();
        expect(res.body.attachments.length).toBeGreaterThan(0);

        taskId = res.body._id;
    });

    // TEST 2: Görevleri Listeleme
    it('GET /api/tasks - Should get all tasks for user', async () => {
        const res = await request(app)
            .get('/api/tasks')
            .set('Authorization', `Bearer ${token}`);

        expect(res.statusCode).toEqual(200);
        expect(Array.isArray(res.body)).toBeTruthy();
        expect(res.body.length).toBeGreaterThan(0);
    });

    // TEST 3: Görev Güncelleme
    it('PUT /api/tasks/:id - Should update a task', async () => {
        const res = await request(app)
            .put(`/api/tasks/${taskId}`)
            .set('Authorization', `Bearer ${token}`)
            .send({
                status: 'Completed',
                title: 'Updated Title'
            });

        expect(res.statusCode).toEqual(200);
        expect(res.body.status).toBe('Completed');
        expect(res.body.title).toBe('Updated Title');
    });

    // TEST 4: Görev Silme
    it('DELETE /api/tasks/:id - Should delete a task', async () => {
        const res = await request(app)
            .delete(`/api/tasks/${taskId}`)
            .set('Authorization', `Bearer ${token}`);

        expect(res.statusCode).toEqual(200);
    });
});
```

Fig. 18. Task test scenarios

B. *Test Results*

All backend API endpoints were validated through automated integration tests. Running the suite with the `npm test` command produced a full pass outcome (all test suites and test cases passed). As shown in Fig. 19, the Authentication and Task modules—together with the multipart file upload scenario—successfully met all assertions, indicating that the backend logic and route protections behave correctly before frontend integration.



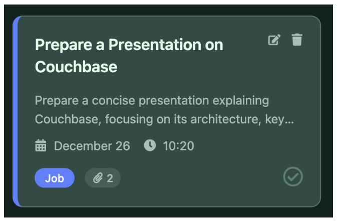
```
senadulger@Sena's-MacbookAir backend % npm test

> backend@1.0.0 test
> jest --detectOpenHandles

 PASS  __tests__/task.test.js
  ● Console

    console.log
      [dotenv@17.2.3] injecting env (2) from .env -- tip: ⚙ suppress all logs with { quiet: true }

      at _log (node_modules/dotenv/lib/main.js:142:11)

    console.log
      MongoDB connected: ac-hzm1hmu-shard-00-00.el53fa1.mongodb.net

      at log (config/db.js:7:13)

    console.log
      Task created successfully with ID: new ObjectId('694938e483faa350f02646aa')

      at log (controllers/taskController.js:98:13)

    console.log
      updateTask called

      at log (controllers/taskController.js:114:11)

    console.log
      body: { status: 'Completed', title: 'Updated Title' }

      at log (controllers/taskController.js:115:11)

    console.log
      files: undefined

      at log (controllers/taskController.js:116:11)

 PASS  __tests__/auth.test.js
  ● Console

    console.log
      [dotenv@17.2.3] injecting env (2) from .env -- tip: 👥 sync secrets across teammates & machines: https://dotenvx.com/ops

      at _log (node_modules/dotenv/lib/main.js:142:11)

    console.log
      MongoDB connected: ac-hzm1hmu-shard-00-00.el53fa1.mongodb.net

      at log (config/db.js:7:13)


Test Suites: 2 passed, 2 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        6.015 s
Ran all test suites.
```

Fig. 19. Test execution output

IX. UI Feature Evidence

A. *Completed Tasks View*

When the checkmark icon at the bottom-right of a task card is clicked, the UI marks the task as completed by turning the card green. At the same time, the task record in MongoDB is updated by setting `status` to `Complete`.

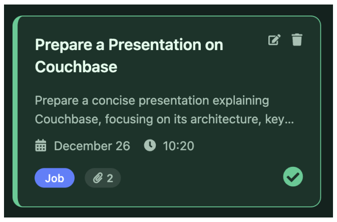


Prepare a Presentation on Couchbase

Prepare a concise presentation explaining Couchbase, focusing on its architecture, key...

December 26 10:20

Fig. 20. Incomplete task card

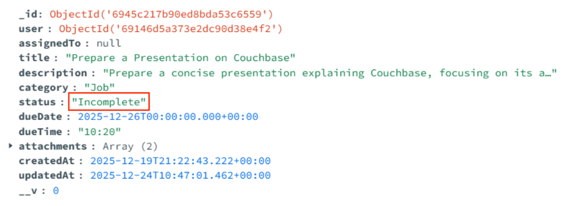


Prepare a Presentation on Couchbase

Prepare a concise presentation explaining Couchbase, focusing on its architecture, key...

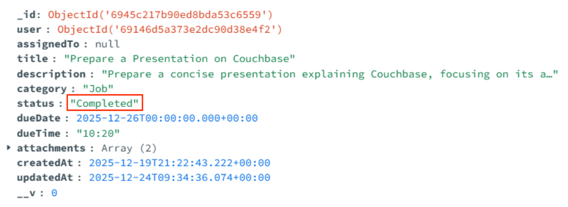
December 26 10:20

Fig. 21. Completed task card



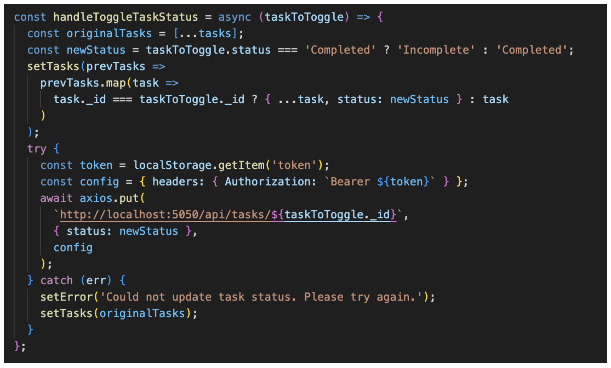
```
  _id: ObjectId('6945c217b90ed8bda53c6559')
  user : ObjectId('69146d5a373e2dc90d38e4f2')
  assignedTo : null
  title : "Prepare a Presentation on Couchbase"
  description : "Prepare a concise presentation explaining Couchbase, focusing on its a…"
  category : "Job"
  status : "Incomplete"
  dueDate : 2025-12-26T00:00:00.000+00:00
  dueTime : "10:20"
▸ attachments : Array (2)
  createdAt : 2025-12-19T21:22:43.222+00:00
  updatedAt : 2025-12-24T10:47:01.462+00:00
  __v : 0
```

Fig. 22. Incomplete task card (DB)



```
  _id: ObjectId('6945c217b90ed8bda53c6559')
  user : ObjectId('69146d5a373e2dc90d38e4f2')
  assignedTo : null
  title : "Prepare a Presentation on Couchbase"
  description : "Prepare a concise presentation explaining Couchbase, focusing on its a…"
  category : "Job"
  status : "Completed"
  dueDate : 2025-12-26T00:00:00.000+00:00
  dueTime : "10:20"
▸ attachments : Array (2)
  createdAt : 2025-12-19T21:22:43.222+00:00
  updatedAt : 2025-12-24T09:34:36.074+00:00
  __v : 0
```

Fig. 23. Completed task card (DB)

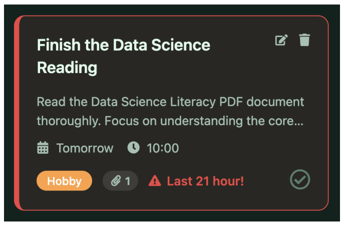


```
const handleToggleTaskStatus = async (taskToToggle) => {
  const originalTasks = [...tasks];
  const newStatus = taskToToggle.status === 'Completed' ? 'Incomplete' : 'Completed';
  setTasks(prevTasks =>
    prevTasks.map(task =>
      task._id === taskToToggle._id ? { ...task, status: newStatus } : task
    )
  );
  try {
    const token = localStorage.getItem('token');
    const config = { headers: { Authorization: `Bearer ${token}` } };
    await axios.put(
      `http://localhost:5050/api/tasks/${taskToToggle._id}`,
      { status: newStatus },
      config
    );
  } catch (err) {
    setError('Could not update task status. Please try again.');
    setTasks(originalTasks);
  }
};
```

Fig. 24. Task status update code

B. *Approaching Deadlines View*

When a task has less than 24 hours remaining until its due date, the task card is highlighted in red. The interface also calculates the remaining time dynamically and displays a warning under the card showing how many hours are left.

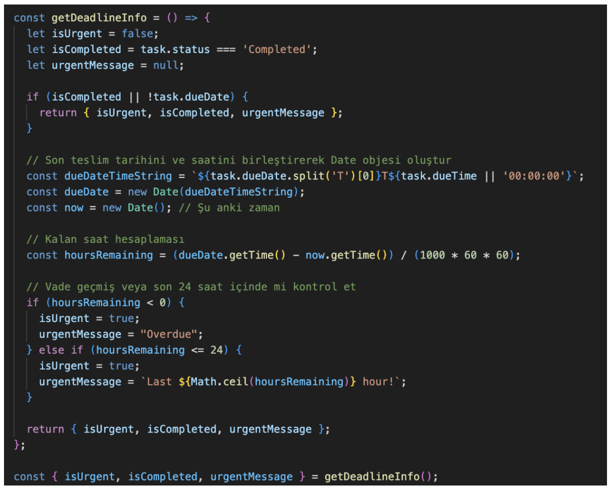


Finish the Data Science Reading

Read the Data Science Literacy PDF document thoroughly. Focus on understanding the core...

Tomorrow 10:00

Fig. 25. Deadline alert (UI)

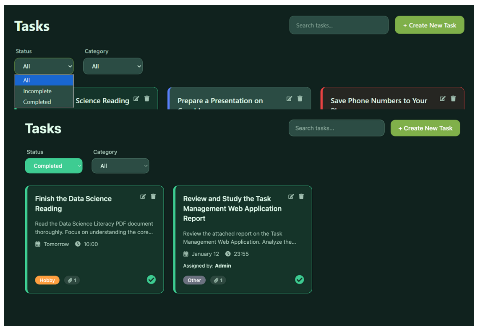


```
const getDeadlineInfo = () => {
  let isUrgent = false;
  let isCompleted = task.status === 'Completed';
  let urgentMessage = null;

  if (isCompleted || !task.dueDate) {
    return { isUrgent, isCompleted, urgentMessage };
  }

  // Son teslim tarihini ve saatini birleştirerek Date objesi oluştur
  const dueDateTimeString = `${task.dueDate.split('T')[0]}T${task.dueTime || '00:00:00'}`;
  const dueDate = new Date(dueDateTimeString);
  const now = new Date(); // Şu anki zaman

  // Kalan saat hesaplaması
  const hoursRemaining = (dueDate.getTime() - now.getTime()) / (1000 * 60 * 60);

  // Vade geçmiş veya son 24 saat içinde mi kontrol et
  if (hoursRemaining < 0) {
    isUrgent = true;
    urgentMessage = "Overdue";
  } else if (hoursRemaining <= 24) {
    isUrgent = true;
    urgentMessage = `Last ${Math.ceil(hoursRemaining)} hour!`;
  }

  return { isUrgent, isCompleted, urgentMessage };
};

const { isUrgent, isCompleted, urgentMessage } = getDeadlineInfo();
```

Fig. 26. Deadline alert (Code)

C. *Task Filtering Feature*

To improve usability, the task list supports filtering by *Status* and *Category* for all users. Admin users additionally have a *User* filter to view tasks by owner. Both roles also include a search bar to quickly find tasks by keywords.



Finish the Data Science Reading

Read the Data Science Literacy PDF document thoroughly. Focus on understanding the core...

Tomorrow 10:00

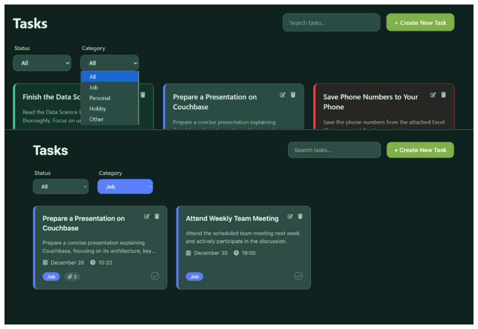
Review and Study the Task Management Web Application Report

Review the attached report on the Task Management Web Application. Analyze the...

January 12 23:55

Assigned by: **Admin**

Fig. 27. Status Filter + Result



Prepare a Presentation on Couchbase

Prepare a concise presentation explaining Couchbase, focusing on its architecture, key...

December 26 10:20

Attend Weekly Team Meeting

Attend the scheduled team meeting next week and actively participate in the discussion.

December 30 19:00

Fig. 28. Category Filter + Result

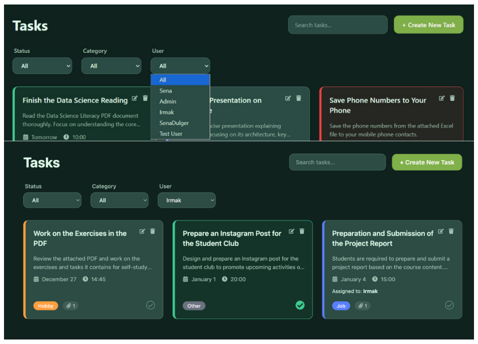

Tasks
Search tasks...
+ Create New Task
Status
Category
User
All
Sena
Admin
Irmak
SenaDulger
Test User
Finish the Data Science Reading
Presentation on
Save Phone Numbers to Your Phone
Work on the Exercises in the PDF
Prepare an Instagram Post for the Student Club
Preparation and Submission of the Project Report
Hobby
Other
Job

Fig. 29. User Filter + Result (Admin)

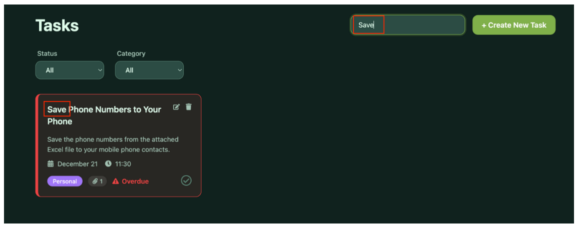
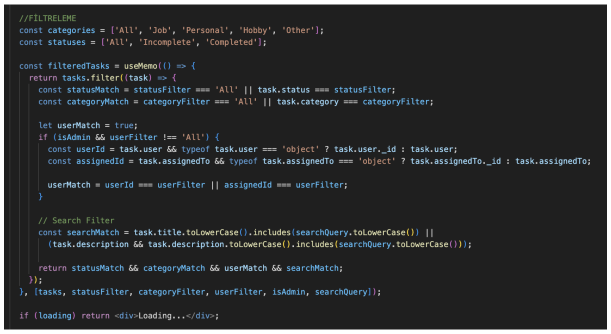

Tasks
Save
+ Create New Task
Status
Category
All
Save Phone Numbers to Your Phone
Save the phone numbers from the attached Excel file to your mobile phone contacts.
December 21
11:30
Personal
Overdue

Fig. 30. Search Bar + Result



```
//FİLTRELEME
const categories = ['All', 'Job', 'Personal', 'Hobby', 'Other'];
const statuses = ['All', 'Incomplete', 'Completed'];

const filteredTasks = useMemo(() => {
  return tasks.filter((task) => {
    const statusMatch = statusFilter === 'All' || task.status === statusFilter;
    const categoryMatch = categoryFilter === 'All' || task.category === categoryFilter;

    let userMatch = true;
    if (isAdmin && userFilter !== 'All') {
      const userId = task.user && typeof task.user === 'object' ? task.user._id : task.user;
      const assignedId = task.assignedTo && typeof task.assignedTo === 'object' ? task.assignedTo._id : task.assignedTo;

      userMatch = userId === userFilter || assignedId === userFilter;
    }

    // Search Filter
    const searchMatch = task.title.toLowerCase().includes(searchQuery.toLowerCase()) ||
      (task.description && task.description.toLowerCase().includes(searchQuery.toLowerCase()));

    return statusMatch && categoryMatch && userMatch && searchMatch;
  });
}, [tasks, statusFilter, categoryFilter, userFilter, isAdmin, searchQuery]);

if (loading) return <div>Loading...</div>;
```

Fig. 31. Filter Code

D. *Task Distribution Analytics*

On the Analysis page, task distribution is visualized in a chart. By clicking the *Export Graph* button, the displayed graph can be downloaded to the computer in PNG format.

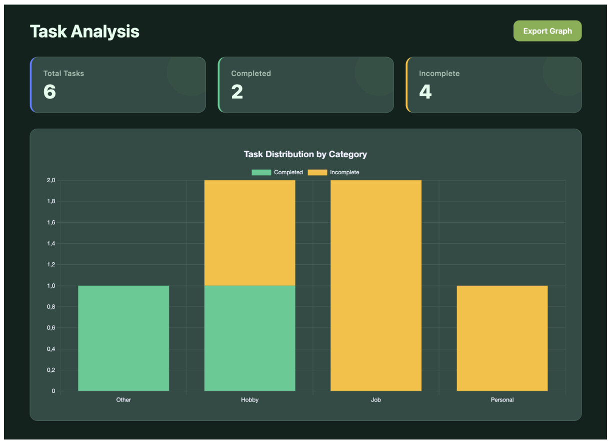


Fig. 32. Analysis page: task distribution chart (web view)

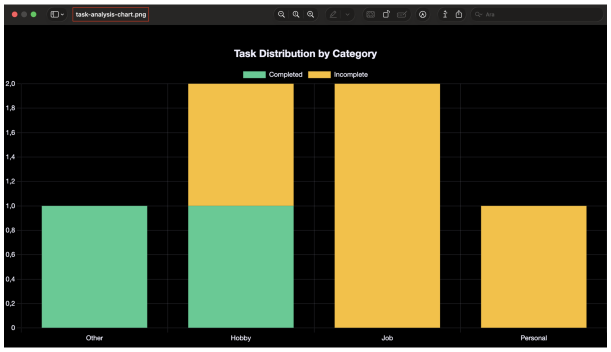
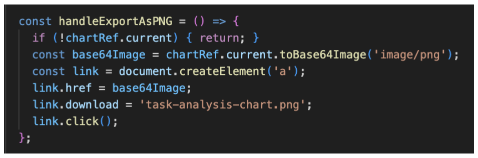


Fig. 33. Export Graph: downloaded chart saved as PNG



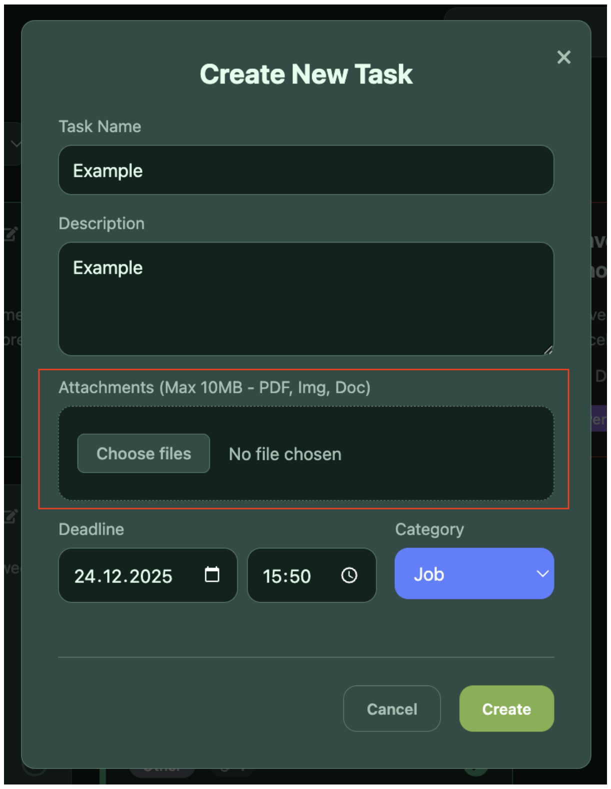
```
const handleExportAsPNG = () => {
  if (!chartRef.current) { return; }
  const base64Image = chartRef.current.toBase64Image('image/png');
  const link = document.createElement('a');
  link.href = base64Image;
  link.download = 'task-analysis-chart.png';
  link.click();
};
```

Fig. 34. Code: PNG export implementation for the chart

X. Extended Requirements

A. *Task-Based File Management*

During task creation, users can attach files up to a maximum size of 10 MB. Supported formats include PDF, PNG, JPG, DOCX, and XLSX. After upload, attachments can be viewed, downloaded to the computer, or removed when no longer needed.



Create New Task

Task Name: Example

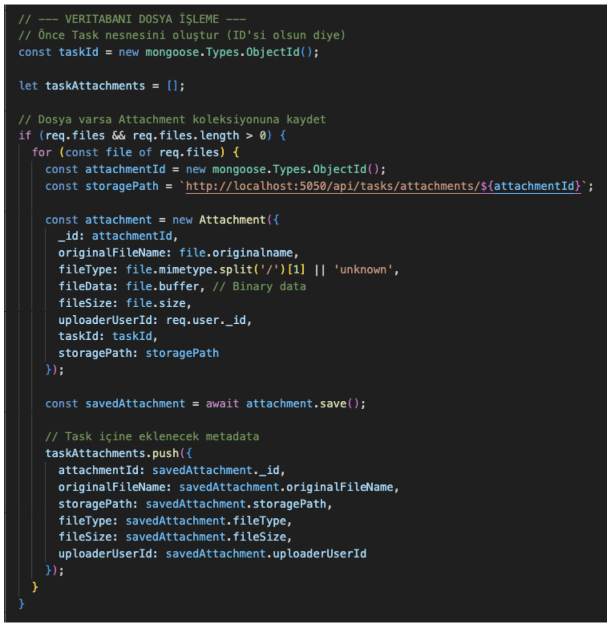
Description: Example

Attachments (Max 10MB - PDF, Img, Doc): Choose files No file chosen

Deadline: 24.12.2025 15:50

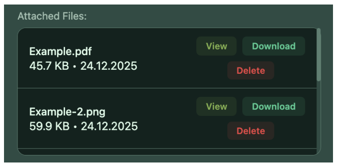
Category: Job

Fig. 35. Task card view



```
// --- VERITABANI DOSYA İŞLEME ---
// Önce Task nesnesini oluştur (ID'si olsun diye)
const taskId = new mongoose.Types.ObjectId();

let taskAttachments = [];

// Dosya varsa Attachment koleksiyonuna kaydet
if (req.files && req.files.length > 0) {
  for (const file of req.files) {
    const attachmentId = new mongoose.Types.ObjectId();
    const storagePath = `http://localhost:5050/api/tasks/attachments/${attachmentId}`;

    const attachment = new Attachment({
      _id: attachmentId,
      originalFileName: file.originalname,
      fileType: file.mimetype.split('/')[1] || 'unknown',
      fileData: file.buffer, // Binary data
      fileSize: file.size,
      uploaderUserId: req.user._id,
      taskId: taskId,
      storagePath: storagePath
    });

    const savedAttachment = await attachment.save();

    // Task içine eklenecek metadata
    taskAttachments.push({
      attachmentId: savedAttachment._id,
      originalFileName: savedAttachment.originalFileName,
      storagePath: savedAttachment.storagePath,
      fileType: savedAttachment.fileType,
      fileSize: savedAttachment.fileSize,
      uploaderUserId: savedAttachment.uploaderUserId
    });
  }
}
```

Fig. 36. File Upload Code

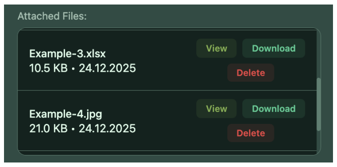


Attached Files:

Example.pdf
45.7 KB • 24.12.2025

Example-2.png
59.9 KB • 24.12.2025

Fig. 37. Attach file UI (PDF, PNG)

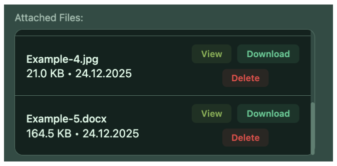


Attached Files:

Example-3.xlsx
10.5 KB • 24.12.2025

Example-4.jpg
21.0 KB • 24.12.2025

Fig. 38. Attach file UI (XLSX, JPG)



Attached Files:

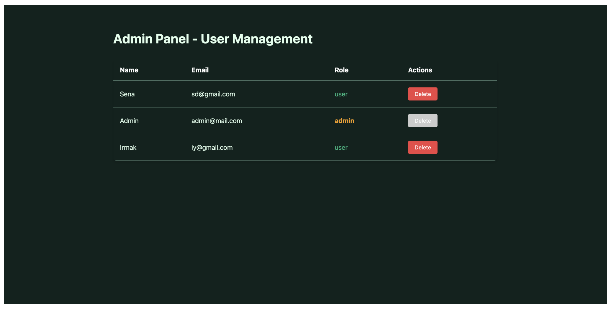
Example-4.jpg
21.0 KB • 24.12.2025

Example-5.docx
164.5 KB • 24.12.2025

Fig. 39. Attach file UI (DOCX)

B. Role-Based Authorization

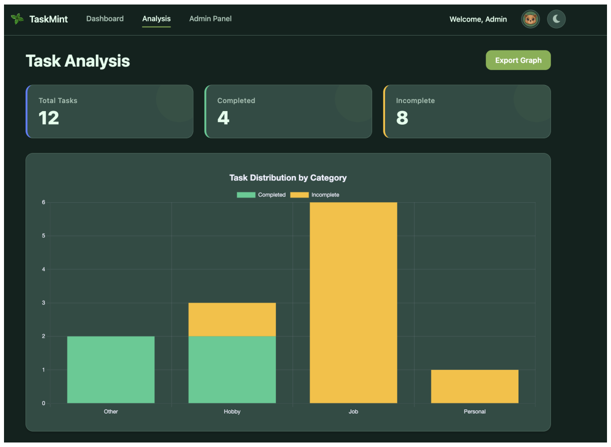
Role-based access control (RBAC) is applied to distinguish admin capabilities. Admin users can view all tasks, assign tasks to any user, and update or delete any task. They can also access the full user list and remove user accounts when needed; however, admin accounts cannot be deleted through the system.



Admin Panel - User Management

Name	Email	Role	Actions
Sena	sd@gmail.com	user	Delete
Admin	admin@mail.com	admin	Delete
Irmak	iy@gmail.com	user	Delete

Fig. 40. Admin User Management



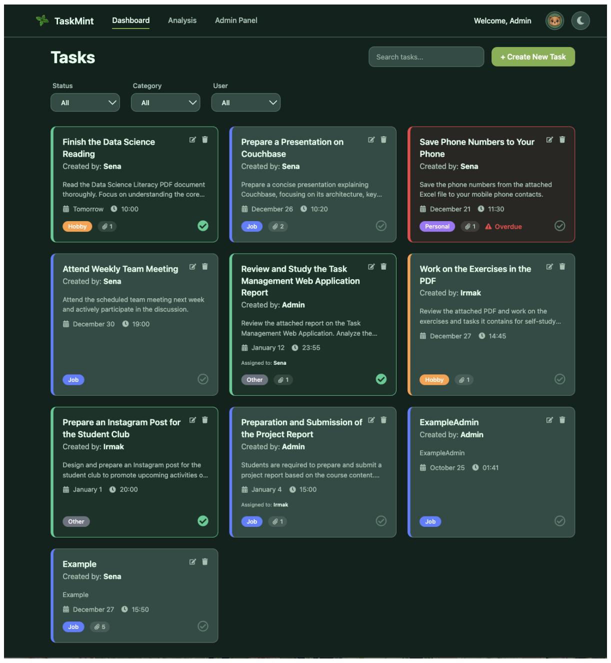
Task Analysis

Total Tasks
12

Completed
4

Incomplete
8

Fig. 41. Admin Analysis View



Tasks

Status
All
Category
All
User
All

Finish the Data Science Reading
Created by: **Sena**
Read the Data Science Literacy PDF document thoroughly. Focus on understanding the core...
Tomorrow 10:00
Hobby 1

Prepare a Presentation on Couchbase
Created by: **Sena**
Prepare a concise presentation explaining Couchbase, focusing on its architecture, key...
December 26 10:20
Job 2

Save Phone Numbers to Your Phone
Created by: **Sena**
Save the phone numbers from the attached Excel file to your mobile phone contacts.
December 21 11:30
Personal 1 Overdue

Attend Weekly Team Meeting
Created by: **Sena**
Attend the scheduled team meeting next week and actively participate in the discussion.
December 30 19:00
Job

Review and Study the Task Management Web Application Report
Created by: **Admin**
Review the attached report on the Task Management Web Application. Analyze the...
January 12 23:55
Assigned to: **Sena**
Other 1

Work on the Exercises in the PDF
Created by: **Irmak**
Review the attached PDF and work on the exercises and tasks it contains for self-study...
December 27 14:45
Hobby 1

Prepare an Instagram Post for the Student Club
Created by: **Irmak**
Design and prepare an Instagram post for the student club to promote upcoming activities o...
January 1 20:00
Other

Preparation and Submission of the Project Report
Created by: **Admin**
Students are required to prepare and submit a project report based on the course content....
January 4 15:00
Assigned to: **Irmak**
Job 1

ExampleAdmin
Created by: **Admin**
ExampleAdmin
October 25 01:41
Job

Example
Created by: **Sena**
Example
December 27 15:50
Job 5

Fig. 42. Admin Dashboard View

XI. Conclusion - 2

All planned development phases for the Task Management Web Application have been successfully completed. The application has evolved from a basic prototype into a fully functional system supporting comprehensive CRUD operations, integrated with advanced features such as task-based file management and role-based access control (RBAC). System stability and reliability were verified through automated backend integration tests using Jest and Supertest, achieving a 100% pass rate. Furthermore, the user interface was finalized with the implementation of the "Analysis" dashboard for data visualization and dynamic visual cues for task status and deadlines. The project now meets all specified requirements, delivering a secure and scalable solution for personal task management.