Darbo skelbimų portalas

Sprendžiamo uždavinio aprašymas:

Šiuolaikinėje darbo rinkoje egzistuoja daugybė darbo skelbimų svetainių, tačiau daugelis jų yra sudėtingos naudoti arba nepateikia galimybės efektyviai filtruoti skelbimus pagal kategorijas. Taip pat trūksta platformų, kuriose naudotojai galėtų diskutuoti ir keistis nuomonėmis apie darbo pasiūlymus.

Sistemos paskirtis:

Sukurti patogią ir intuityvią platformą, leidžiančią naudotojams:

- Peržiūrėti darbo skelbimus pagal tam tikras kategorijas.
- Skaityti ir pridėti komentarus prie konkrečių darbo skelbimų.
- Talpinti darbo skelbimus ir redaguoti juos.

Sistema bus skirta trijų tipų naudotojams: svečiams, registruotiems naudotojams ir administratoriams, suteikiant kiekvienai rolei atitinkamas funkcijas ir teises.

Funkciniai reikalavimai:

1. Svečias:
    - Gali peržiūrėti skelbimų kategorijas ir sąrašus.
    - Gali skaityti komentarus prie skelbimų.
    - Neturi galimybės pridėti komentarų ar skelbimų.

2. Registruotas naudotojas:
    - Visos svečio teisės.
    - Gali registruotis ir prisijungti prie sistemos.
    - Gali pridėti komentarus prie darbo skelbimų.
    - Gali talpinti ir redaguoti savo darbo skelbimus.
    - Gali redaguoti arba trinti savo komentarus.

3. Administratorius:
    - Visos registruoto naudotojo teisės.
    - Gali pridėti, redaguoti ir trinti darbo skelbimus, įskaitant kitų naudotojų skelbimus.
    - Gali kurti ir tvarkyti skelbimų kategorijas.
    - Gali moderuoti komentarus: redaguoti arba trinti netinkamus komentarus.

Pasirinktų technologijų aprašymas:

- Front-end:
  - o React.js: moderniai ir dinamiškai naudotojo sąsajai kurti.

- Back-end:
  - o Node.js su Express.js: serverio pusės logikai ir API kūrimui.
  - o Duomenų bazė MongoDB: darbo skelbimų, naudotojų ir komentarų duomenų saugojimui.

- Versijų kontrolė:
  - o GitHub: kodo saugyklai ir projektų valdymui.

OpenAPI specifikacija:

```
openapi: 3.0.0
info:
 title: Job Portal API
 description: API to manage users, jobs, categories, and comments.
 version: 1.0.0

servers:
 - url: http://localhost:5000

paths:
 /users:
  post:
   summary: Register a new user
   operationId: registerUser
   requestBody:
    description: User data to register
    required: true
    content:
     application/json:
      schema:
       type: object
       required:
        - name
        - email
```

```
          - password
        properties:
          name:
            type: string
            example: John Doe
          email:
            type: string
            format: email
            example: johndoe@example.com
          password:
            type: string
            minLength: 7
            example: password123
    responses:
      '201':
        description: User registered successfully
        content:
          application/json:
            schema:
              type: object
              properties:
                _id:
                  type: string
                name:
                  type: string
                email:
                  type: string
                role:
                  type: string
      '422':
        description: Password too short
      '400':
        description: Bad request

  /users/{id}:
    get:
      summary: Get a user by ID
      operationId: getUserById
      parameters:
        - name: id
          in: path
```

```
    required: true
    schema:
     type: string
  responses:
   '200':
    description: User information
    content:
     application/json:
      schema:
       type: object
       properties:
        _id:
         type: string
        name:
         type: string
        email:
         type: string
        role:
         type: string
   '404':
    description: User not found

put:
  summary: Update user information
  operationId: updateUser
  parameters:
   - name: id
    in: path
    required: true
    schema:
     type: string
  requestBody:
   description: Updated user data
   required: true
   content:
    application/json:
     schema:
      type: object
      properties:
       name:
        type: string
```

```
          example: John Doe
        email:
          type: string
          example: johndoe@example.com
  responses:
   '200':
    description: User updated successfully
    content:
     application/json:
      schema:
       type: object
   '404':
    description: User not found

/categories:
 get:
  summary: Get all categories
  operationId: getAllCategories
  responses:
   '200':
    description: List of categories
    content:
     application/json:
      schema:
       type: array
       items:
        type: object
        properties:
         _id:
          type: string
         name:
          type: string

 post:
  summary: Create a new category
  operationId: createCategory
  requestBody:
   description: Category data to create
   required: true
   content:
    application/json:
```

```
        schema:
          type: object
          required:
            - name
          properties:
            name:
              type: string
              example: IT
    responses:
      '201':
        description: Category created successfully
        content:
          application/json:
            schema:
              type: object
      '400':
        description: Bad request

  /categories/{id}:
    put:
      summary: Update a category
      operationId: updateCategory
      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: string
      requestBody:
        description: Updated category data
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                name:
                  type: string
                  example: New Category Name
      responses:
        '200':
```

```
      description: Category updated successfully
      content:
       application/json:
        schema:
         type: object
    '404':
      description: Category not found

  delete:
   summary: Delete a category by ID
   operationId: deleteCategory
   parameters:
    - name: id
      in: path
      required: true
      schema:
       type: string
   responses:
    '204':
      description: Category deleted successfully
    '404':
      description: Category not found

/categories/{id}/jobs:
  get:
   summary: Get jobs by category
   operationId: getJobsByCategory
   parameters:
    - name: id
      in: path
      required: true
      schema:
       type: string
   responses:
    '200':
      description: List of jobs by category
      content:
       application/json:
        schema:
         type: array
         items:
```

```
            type: object
            properties:
              _id:
                type: string
              title:
                type: string
              description:
                type: string
              categoryId:
                type: string
    '404':
      description: Category not found

/jobs:
  get:
    summary: Get all jobs
    operationId: getAllJobs
    responses:
      '200':
        description: List of jobs
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  _id:
                    type: string
                  title:
                    type: string
                  description:
                    type: string
                  categoryId:
                    type: string
                  userId:
                    type: string

  post:
    summary: Create a new job
    operationId: createJob
```

```
    requestBody:
      description: Job data to create
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - title
              - description
              - categoryId
              - userId
            properties:
              title:
                type: string
                example: Senior Software Engineer
              description:
                type: string
                example: Looking for a senior software engineer.
              categoryId:
                type: string
                example: 60da04c0b1a627001ce8d6fa
              userId:
                type: string
                example: 60d9f8c7f1b3d45b5c0567e5
    responses:
      '201':
        description: Job created successfully
        content:
          application/json:
            schema:
              type: object
      '400':
        description: Bad request

  /jobs/{id}:
    get:
      summary: Get a job by ID
      operationId: getJobById
      parameters:
        - name: id
```

```
    in: path
    required: true
    schema:
     type: string
  responses:
   '200':
    description: Job information
    content:
     application/json:
      schema:
       type: object
   '404':
    description: Job not found

 put:
  summary: Update a job by ID
  operationId: updateJob
  parameters:
   - name: id
    in: path
    required: true
    schema:
     type: string
  requestBody:
   description: Updated job data
   required: true
   content:
    application/json:
     schema:
      type: object
      properties:
       title:
        type: string
        example: Senior Developer
       description:
        type: string
        example: Updated job description.
  responses:
   '200':
    description: Job updated successfully
    content:
```

```
        application/json:
          schema:
            type: object
    '404':
      description: Job not found

  delete:
    summary: Delete a job by ID
    operationId: deleteJob
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: string
    responses:
      '204':
        description: Job deleted successfully
      '404':
        description: Job not found

/jobs/{jobId}/comments:
  get:
    summary: Get comments for a job
    operationId: getCommentsForJob
    parameters:
      - name: jobId
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: List of comments
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
```

```
            _id:
              type: string
            content:
              type: string
            jobId:
              type: string
            userId:
              type: string

post:
  summary: Add a comment to a job
  operationId: addCommentToJob
  parameters:
    - name: jobId
      in: path
      required: true
      schema:
        type: string
  requestBody:
    description: Comment data to add
    required: true
    content:
      application/json:
        schema:
          type: object
          required:
            - content
            - userId
          properties:
            content:
              type: string
              example: This job looks interesting.
            userId:
              type: string
              example: 60d9f8c7f1b3d45b5c0567e5
  responses:
    '201':
      description: Comment added successfully
      content:
        application/json:
          schema:
```

```
          type: object

/comments/{id}:
  get:
    summary: Get a comment by ID
    operationId: getCommentById
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Comment information
        content:
          application/json:
            schema:
              type: object
      '404':
        description: Comment not found

  put:
    summary: Update a comment by ID
    operationId: updateComment
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: string
    requestBody:
      description: Updated comment data
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              content:
                type: string
```

```
          example: Updated comment content
  responses:
   '200':
    description: Comment updated successfully
    content:
     application/json:
      schema:
       type: object
   '404':
    description: Comment not found

delete:
  summary: Delete a comment by ID
  operationId: deleteComment
  parameters:
   - name: id
    in: path
    required: true
    schema:
     type: string
  responses:
   '204':
    description: Comment deleted successfully
   '404':
    description: Comment not found
```