**Data Models**

**1. User Model**

**Description:** Represents a registered user of the platform.

**Suggested Fields:**

* **id** (string | UUID): Unique identifier for the user.
* **username** (string, required, unique): Chosen username.
* **name** (string, optional): The user’s real name.
* **bio** (string, optional): A short bio describing the user.
* **skills** (array of strings, optional): List of programming skills, e.g. ["JavaScript", "Python"].
* **githubProfileUrl** (string, optional): Link to GitHub profile.
* **gitlabProfileUrl** (string, optional): Link to GitLab profile.
* **profileImageUrl** (string, optional): URL to the user’s profile picture.
* **createdAt** (datetime): Timestamp when the user was created.
* **updatedAt** (datetime): Timestamp when the user was last updated.

**Indices:**

* Primary: id
* Unique: username

**Relationships:**

* A User can have many Snippets.
* A User can have many Comments.
* A User can participate in many Projects (through a relationship table if needed).

**2. Snippet Model**

**Description:** Represents a code snippet shared by a user.

**Suggested Fields:**

* **id** (string | UUID): Unique identifier for the snippet.
* **ownerId** (string | UUID, required): The ID of the user who created the snippet.
* **title** (string, required): A short title for the snippet.
* **description** (string, optional): Detailed description of what the snippet does.
* **code** (string, required): The actual code content.
* **language** (string, required): Programming language (e.g. "JavaScript", "Python", "Go").
* **tags** (array of strings, optional): Tags for categorization (e.g. ["frontend", "authentication"]).
* **createdAt** (datetime): Timestamp of creation.
* **updatedAt** (datetime): Timestamp of last update.

**Indices:**

* Primary: id
* Index: ownerId, tags, language

**Relationships:**

* A Snippet belongs to a User (owner).
* A Snippet has many Comments.

**3. Comment Model**

**Description:** Represents a user’s comment on a snippet.

**Suggested Fields:**

* **id** (string | UUID): Unique identifier for the comment.
* **snippetId** (string | UUID, required): The ID of the snippet the comment is on.
* **authorId** (string | UUID, required): The ID of the user who posted the comment.
* **content** (string, required): The comment text.
* **createdAt** (datetime): Timestamp of creation.
* **updatedAt** (datetime): Timestamp of last update.

**Indices:**

* Primary: id
* Index: snippetId, authorId

**Relationships:**

* A Comment belongs to a Snippet.
* A Comment is authored by a User.

**4. Project Model**

**Description:** Represents a collaborative project that users can join and share code repositories.

**Suggested Fields:**

* **id** (string | UUID): Unique identifier for the project.
* **ownerId** (string | UUID, required): The ID of the user who created the project.
* **name** (string, required): Project name.
* **description** (string, optional): Brief description of the project.
* **repoUrl** (string, optional): URL to the main code repository (e.g. GitHub link).
* **createdAt** (datetime): Timestamp of creation.
* **updatedAt** (datetime): Timestamp of last update.

**Indices:**

* Primary: id
* Index: ownerId

**Relationships:**

* A Project belongs to a User (owner).
* A Project has many members (Users) who collaborate on it.  
  (This may be implemented via a ProjectMembership join table with fields: id, projectId, userId, role.)

**5. ProjectMembership Model (If needed)**

**Description:** Connects users and projects, indicating who is a member of which project.

**Suggested Fields:**

* **id** (string | UUID): Unique identifier.
* **projectId** (string | UUID, required): The ID of the project.
* **userId** (string | UUID, required): The ID of the user who is a member.
* **role** (string, optional): Role in the project, e.g. ["maintainer", "contributor"].
* **createdAt** (datetime)
* **updatedAt** (datetime)

**Indices:**

* Primary: id
* Unique compound index: (projectId, userId)

**API Endpoints**

Base URL: https://api.codeconnect.com/v1 (for example)

All endpoints could return standard HTTP status codes:

* 200 OK for successful GET/PUT/POST requests.
* 201 Created for successful resource creation.
* 204 No Content for successful deletions.
* 400 Bad Request for invalid input.
* 401 Unauthorized if the user is not authenticated.
* 403 Forbidden if the user is not authorized to access the resource.
* 404 Not Found if the requested resource is not found.
* 500 Internal Server Error for unexpected errors.

**/users**

**POST /users**

* **Description:** Create a new user account.
* **Request Body (JSON):**

json

Copy code

{

"username": "john\_doe",

"name": "John Doe",

"bio": "Full-stack developer.",

"skills": ["JavaScript", "React"],

"githubProfileUrl": "https://github.com/john\_doe",

"gitlabProfileUrl": "https://gitlab.com/john\_doe",

"profileImageUrl": "https://example.com/images/john.jpg"

}

* **Response (201 Created):**

json

Copy code

{

"id": "user-uuid",

"username": "john\_doe",

"name": "John Doe",

"bio": "Full-stack developer.",

"skills": ["JavaScript", "React"],

"githubProfileUrl": "https://github.com/john\_doe",

"gitlabProfileUrl": "https://gitlab.com/john\_doe",

"profileImageUrl": "https://example.com/images/john.jpg",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

**GET /users/{userId}**

* **Description:** Retrieve user profile details.
* **Response (200 OK):**

json

Copy code

{

"id": "user-uuid",

"username": "john\_doe",

"name": "John Doe",

"bio": "Full-stack developer.",

"skills": ["JavaScript", "React"],

"githubProfileUrl": "https://github.com/john\_doe",

"gitlabProfileUrl": "https://gitlab.com/john\_doe",

"profileImageUrl": "https://example.com/images/john.jpg",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

**PUT /users/{userId}**

* **Description:** Update user profile information.
* **Request Body (JSON):** (All fields optional)

json

Copy code

{

"name": "Johnathan Doe",

"bio": "Experienced full-stack developer.",

"skills": ["JavaScript", "React", "Node.js"],

"profileImageUrl": "https://example.com/images/john-updated.jpg"

}

* **Response (200 OK):** Returns the updated user object.

json

Copy code

{

"id": "user-uuid",

"username": "john\_doe",

"name": "Johnathan Doe",

"bio": "Experienced full-stack developer.",

"skills": ["JavaScript", "React", "Node.js"],

"githubProfileUrl": "https://github.com/john\_doe",

"gitlabProfileUrl": "https://gitlab.com/john\_doe",

"profileImageUrl": "https://example.com/images/john-updated.jpg",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T11:00:00Z"

}

**/snippets**

**POST /snippets**

* **Description:** Create a new code snippet.
* **Request Body (JSON):**

json

Copy code

{

"ownerId": "user-uuid",

"title": "Quick Sort Example",

"description": "A quick sort implementation in JavaScript",

"code": "function quickSort(arr) { ... }",

"language": "JavaScript",

"tags": ["sorting", "algorithm"]

}

* **Response (201 Created):**

json

Copy code

{

"id": "snippet-uuid",

"ownerId": "user-uuid",

"title": "Quick Sort Example",

"description": "A quick sort implementation in JavaScript",

"code": "function quickSort(arr) { ... }",

"language": "JavaScript",

"tags": ["sorting", "algorithm"],

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

**GET /snippets/{snippetId}**

* **Description:** Retrieve a specific code snippet.
* **Response (200 OK):**

json

Copy code

{

"id": "snippet-uuid",

"ownerId": "user-uuid",

"title": "Quick Sort Example",

"description": "A quick sort implementation in JavaScript",

"code": "function quickSort(arr) { ... }",

"language": "JavaScript",

"tags": ["sorting", "algorithm"],

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

**PUT /snippets/{snippetId}**

* **Description:** Update an existing code snippet.
* **Request Body (JSON):** (Any field optional)

json

Copy code

{

"title": "Quick Sort (Optimized)",

"description": "An optimized quick sort example",

"code": "function quickSort(arr) { /\* optimized code \*/ }",

"tags": ["sorting", "algorithm", "performance"]

}

* **Response (200 OK):**

json

Copy code

{

"id": "snippet-uuid",

"ownerId": "user-uuid",

"title": "Quick Sort (Optimized)",

"description": "An optimized quick sort example",

"code": "function quickSort(arr) { /\* optimized code \*/ }",

"language": "JavaScript",

"tags": ["sorting", "algorithm", "performance"],

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T11:00:00Z"

}

**DELETE /snippets/{snippetId}**

* **Description:** Delete a code snippet.
* **Response (204 No Content)**

**GET /snippets**

* **Description:** Search and filter code snippets by user, tag, language.
* **Query Params (optional):** ownerId=..., tags=... (comma-separated), language=..., q=... (for keyword in title/description).
* **Response (200 OK):**

json

Copy code

[

{

"id": "snippet-uuid",

"ownerId": "user-uuid",

"title": "Quick Sort Example",

"description": "A quick sort implementation in JavaScript",

"language": "JavaScript",

"tags": ["sorting", "algorithm"],

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

]

**/comments**

**POST /comments/{snippetId}**

* **Description:** Add a comment to a code snippet.
* **Request Body (JSON):**

json

Copy code

{

"authorId": "user-uuid",

"content": "Great snippet! Have you considered adding a partition function?"

}

* **Response (201 Created):**

json

Copy code

{

"id": "comment-uuid",

"snippetId": "snippet-uuid",

"authorId": "user-uuid",

"content": "Great snippet! Have you considered adding a partition function?",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

**GET /comments/{snippetId}**

* **Description:** Retrieve comments for a code snippet.
* **Response (200 OK):**

json

Copy code

[

{

"id": "comment-uuid",

"snippetId": "snippet-uuid",

"authorId": "user-uuid",

"content": "Great snippet! Have you considered adding a partition function?",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

]

**/projects**

**POST /projects**

* **Description:** Create a new project.
* **Request Body (JSON):**

json

Copy code

{

"ownerId": "user-uuid",

"name": "Awesome Project",

"description": "A collaborative web application.",

"repoUrl": "https://github.com/john\_doe/awesome-project"

}

* **Response (201 Created):**

json

Copy code

{

"id": "project-uuid",

"ownerId": "user-uuid",

"name": "Awesome Project",

"description": "A collaborative web application.",

"repoUrl": "https://github.com/john\_doe/awesome-project",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z"

}

**GET /projects/{projectId}**

* **Description:** Retrieve project details.
* **Response (200 OK):**

json

Copy code

{

"id": "project-uuid",

"ownerId": "user-uuid",

"name": "Awesome Project",

"description": "A collaborative web application.",

"repoUrl": "https://github.com/john\_doe/awesome-project",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T10:00:00Z",

"members": [

{

"userId": "user-uuid",

"role": "maintainer"

}

]

}

**PUT /projects/{projectId}**

* **Description:** Update project information.
* **Request Body (JSON):** (All fields optional)

json

Copy code

{

"name": "Awesome Project v2",

"description": "An upgraded collaborative web application.",

"repoUrl": "https://github.com/john\_doe/awesome-project-v2"

}

* **Response (200 OK):**

json

Copy code

{

"id": "project-uuid",

"ownerId": "user-uuid",

"name": "Awesome Project v2",

"description": "An upgraded collaborative web application.",

"repoUrl": "https://github.com/john\_doe/awesome-project-v2",

"createdAt": "2024-12-17T10:00:00Z",

"updatedAt": "2024-12-17T11:00:00Z"

}

**POST /projects/{projectId}/members**

* **Description:** Add a user to a project.
* **Request Body (JSON):**

json

Copy code

{

"userId": "user-uuid",

"role": "contributor"

}

* **Response (201 Created):**

json

Copy code

{

"projectId": "project-uuid",

"userId": "user-uuid",

"role": "contributor",

"createdAt": "2024-12-17T10:00:00Z"

}

**Additional Considerations**

* **Authentication & Authorization:**  
  You might need authentication endpoints (e.g., /auth/login, /auth/signup) and tokens (JWT) to secure your endpoints. Authorization checks would be required to ensure that only owners can update or delete their resources.
* **Pagination, Filtering & Sorting:**  
  For listing endpoints like /snippets or /comments/{snippetId}, consider adding parameters such as page, limit, sortField, sortOrder for improved user experience.
* **Error Handling:**  
  Standardize error responses to include an error object:

json

Copy code

{

"error": {

"message": "Invalid user ID",

"code": 400

}

}

* **Validation:**  
  Validate all inputs to ensure required fields are present and correctly formatted. For example, ensure ownerId is a valid UUID and that title and code are not empty strings.
* **Versioning:**  
  Prefixing the API with /v1 can help manage future backward-incompatible changes.

This comprehensive specification should help guide you in creating prompts for GitHub Copilot. You can ask Copilot to generate boilerplate code (e.g., TypeScript interfaces for these models, database schemas, API route handlers, or even test cases) using the above definitions.