Login	
Endpoint	POST /auth/login
Errors	400 invalid username/password

Request

```
{
   username: string
   password: string
}
```

Response

```
{
   username: string
   email: string
   firstName: string
   lastName: string
   createdAt: date
   updatedAt: date
   sessionAge: int
}
```

Logout	
Endpoint	GET /auth/logout
Errors	No response from server
Request	Response
{ }	{ }

Register	
Endpoint	POST /register
Errors	400 reason for failed registration

Request

```
{
   username: string
   email: string
   password: string
   firstName: string
   lastName: string
}
```

Response

```
{
   msg: string
}
```

Session	
Used to verify whether the user has a valid session. The backend server will examine the cookie in request to decided session validity.	
Endpoint	POST /auth/session
Errors	302 redirect client to /Login

Request

```
{ }
```

Response

```
{
  user:{
      username: string
      email: string
      firstName: string
      lastName: string
      createdAt: date
```

	`updatedAt: date`
	`sessionAge: int`
	`}}`

CreateTeam	
Create a new team. The creator will be team leader and will automatically be a member of the team. Members that do not exist will be ignored. The members array can contain usernames or emails of the users to be include in the team.	
Endpoint	POST /CreateTeam
Errors	400 reason for failed creation

Request	Response

`{`
`    name: string`
`    description: string`
`    members: string[]`
`}`
	`{`
	`  teamId: int`
	`  msg: string`
	`}`

DeleteTeam	
Only team owner can delete a team. Teams can be deleted by name or ID. Therefore, the request should include either teamId or teamName.	
Endpoint	POST /DeleteTeam
Errors	400 reason for failed deletion

Request	Response

`{`
`   teamId/teamName: int/String`
`}`
	`{`
	`  msg: string`
	`}`

GetTeams	
Get a list of all teams. Setting allTeams to true would return all teams including the ones the user is not a member of. Response contains an array of teams. Data won't be sent all at once. Instead, request must specify the limit and offset. Limit indicates the maximum number of records that will be included in the response. For example, limit = 10 and offset = 0 will return the first 10 rows, limit = 10 and offset = 1 will return the second 10 rows (from 10 to 20), and so on.	
Endpoint	POST /GetTeams
Errors	

Request	Response

`{`
`      allTeams: Boolean`
`      paging: {`
`          limit: int`
`          offset: int`
`      }`
`}`
	`{`
	`  numTeams: int`
	`  teams:[{`
	`     id: int`
	`     name: string`
	`     description: string`
	`     createdAt: date`
	`     updatedAt: date`
	`     ownerId: int`

	}] }

GetTeam	
Returns a single team along with team members.	
Endpoint	POST /GetTeam
Errors	400 team not found
Request	Response

{
    teamId: int
}
	{
	  id: int
	  name: string
	  description: string
	  createdAt: date
	  updatedAt: date
	  leaderId: int
	  members: [{
	      id: int
	      username: string
	      email: string
	      firstName: string
	      lastName: string
	      member_since: date
	  }]
	}

CreateProject	
Create a new project. Project creator will be the backlog owner.	
Endpoint	POST /CreateProject
Errors	400 reason for failed creation
Request	Response

{
      name: string
      description: string
      teamId/teamName: int/string
}
	{
	  id: int
	  msg: string
	}

DeleteProject	
The user must be an admin or project owner in order to be able to delete a project.	
Endpoint	POST /DeleteProject
Errors	400 reason for failed deletion
Request	Response

{
projectId/projectName:
int/String
}
	{
	  msg: string
	}

GetProjects	
Get a list of all projects. Setting allProjects to true would return all projects including the ones the user is not working on. Response contains an array of projects. This works the same as GetTeams.	
Endpoint	POST /GetProjects
Errors	

Request

```
{
      allProjects: Boolean
      paging: {
           limit: int
           offset: int
         }
}
```

Response

```
{
  numProjects: int
  projects:[{
     id: int
     name: string
     description: string
     createdAt: date
     updatedAt: date
     teamId: int
     ownerId: int
  }]
}
```

GetProject	
Returns a single project along with its team members and sprints.	
Endpoint	POST /GetProject
Errors	400 project not found

Request

```
{
    projectId: int
}
```

Response

```
{
  id: int
  name: string
  description: string
  createdAt: date
  updatedAt: date
  teamId: int
  ownerId: int
  members: [{
      id: int
      username: string
      email: string
      firstName: string
      lastName: string
  }]
  sprints: [{
      id: int
      description: string
      startDate: date
      endDate: date
      createdAt: date
      updatedAt: date
```

	`}]`
	`}`

CreateTask	
Create a new task (log) under a specified project. Check the statusId and categoryId tables below for more information about which ID to use in the request.	
Endpoint	POST /CreateTask
Errors	400 task already exists

Request

```
{
  log: string
  categoryId: int
  projectId: int
  statusId: int
}
```

Response

```
{
    taskId: int
    msg: string
}
```

statusId		
ID	**Name**	**Description**
20	Created	Newly created task in the project backlog
21	Sprint	Task has been added to a sprint
22	Returned	Task has been returned from a sprint without being completed
23	Completed	Task has been completed

categoryId	
ID	**Name**
10	User Story
11	Bug
12	Refactoring Task

DeleteTask	
Delete a task from the backlog.	
Endpoint	POST /DeleteTask
Errors	400 reason for failed deletion

Request

```
{
    taskId: int
}
```

Response

```
{
  msg: string
}
```

GetTasks
Retrieve tasks. This call can be used in different ways as follow: 1. Retrieve tasks that belong to a certain project. 2. Retrieve tasks that belong to a certain project sprint

Note #1: By default, tasks of all statuses and categories are fetched. To fetch tasks that have certain status/category, supply a status/category property. For example, the following request will retrieve all tasks in projectId 100 that either have the status “Created” (id 20) or “Returned” (id 22) and category “User Story” (id 10):

```
{
  projectId: 100
  status: [20, 22]
  category: [10]
  paging: {
          limit: int
          offset: int
        }
}
```

Note #2: Similar to retrieving teams and projects, **GetTasks** endpoint uses pagination with the default being 10 entries per page.

Endpoint	POST /GetTasks
Errors	400, error “msg”

Request

```
To retrieve tasks in a project:

{
  projectId: int
  status:   int[]     (optional)
  category: int[]     (optional)
  paging: {
          limit: int
          offset: int
        }
}
```

Response

```
{
    numTasks: int
    tasks: [{
       id: int
       name: int
       createdAt: date
       updatedAt: date
       categoryId: int
       categoryName: string
       statusId: int
       statusName: string
    }]
}
```

Request

```
To retrieve tasks in a project
sprint:

{
  ~~projectId: int~~
  sprintId: int
  status:   int[]     (optional)
  category: int[]     (optional)
```

Response

```
{
    numTasks: int
    tasks: [{
       id: int
       name: int
       createdAt: date
       updatedAt: date
       categoryId: int
       categoryName: string
```

```
  paging: {
          limit: int
          offset: int
        }
}
```
	```
	        statusId: int
	        statusName: string
	        estimate: int
	        assigneeId: int
	        assigneeName: string
	     }]
	}
	```

CreateSprint	
Create a sprint for a specified project. If startDate and endDate are omitted, the sprint will have a default of one month starting from the time it’s created.	
Endpoint	POST /CreateSprint
Errors	400

Request	Response

```
{
  description: string
  startDate: date
  endDate: date
  projectId: int
  tasks: [{
     taskId: int
     assigneeId: int
     estimate: int
  }]
}
```
	<pre>
	{
	    sprintId: int
	    msg: string
	}</pre>

DeleteSprint	
Delete a sprint from the backlog.	
Endpoint	POST /DeleteSprint
Errors	400 reason for failed deletion

Request	Response

```
{
    sprintId: int
}
```
	```
	{
	  msg: string
	}
	```

GetSprint	
Two possible usages: 1. Get a single sprint along with its tasks info. Limit and offset determine the set of tasks that will be returned by this endpoint. 2. Get current sprint along with its tasks info. Limit and offset determine the set of tasks that will be returned by this endpoint.	
Endpoint	POST /GetSprint
Errors	400 sprint not found

Request	Response
Usage (1):	

```
{
    sprintId: int
    paging: {
          limit: int
          offset: int
        }
}
```

```
{
  id: int
  description: string
  startDate: date
  endDate: date
  projectId: int
  projectName: string
  numTasks: int
  tasks: [{
     id: int
     name: string
     categoryId: int
     categoryName: string
     statusId: int
     statusName: string
     estimate: int
     assigneeId: int
     assigneeName: string
  }]
}
```

Usage (2):

```
{
    projectId: int
    paging: {
          limit: int
          offset: int
        }
}
```

```
{
  id: int
  description: string
  startDate: date
  endDate: date
  projectId: int
  projectName: string
  numTasks: int
  tasks: [{
     id: int
     name: string
     categoryId: int
     categoryName: string
     statusId: int
     statusName: string
     estimate: int
     assigneeId: int
     assigneeName: string
  }]
}
```