

HackFlow - Complete API Endpoints

Authentication Routes (/api/auth)

Method	Endpoint	Description	Auth Required	Role Required
POST	/register	Register new user	✗	-
POST	/login	Login user	✗	-
POST	/logout	Logout user	✓	-
GET	/me	Get current user details	✓	-
GET	/is-logged-in	Check if user is logged in	✗	-
POST	/refresh-token	Refresh access token	✓	-

User Routes (/api/users)

Method	Endpoint	Description	Auth Required	Role Required
GET	/	Get all users	✓	Admin
GET	/:id	Get user by ID	✓	-
PATCH	/profile	Update own profile	✓	-
PATCH	/:id/role	Update user role	✓	Admin
DELETE	/:id	Delete user	✓	Admin
PATCH	/deactivate	Deactivate own account	✓	-
PATCH	/:id/activate	Activate user account	✓	Admin

Hackathon Routes (/api/hackathons)

Method	Endpoint	Description	Auth Required	Role Required
POST	/create	Create new hackathon	✓	Organizer/Admin
GET	/	Get all hackathons	✗	-
GET	/:id	Get hackathon by ID	✗	-
PATCH	/:id	Update hackathon	✓	Organizer (owner)
DELETE	/:id	Delete hackathon	✓	Organizer (owner)
POST	/:id/join	Join hackathon	✓	Participant
POST	/:id/leave	Leave hackathon	✓	Participant
POST	/:id/assign-judge	Assign judge	✓	Organizer (owner)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/:id/remove-judge</code>	Remove judge	✓	Organizer (owner)

Round Routes (`/api/rounds`)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/create/:hackathonId</code>	Create round	✓	Organizer (owner)
GET	<code>/hackathon/:hackathonId</code>	Get all rounds for hackathon	✓	-
GET	<code>/:id</code>	Get round by ID	✓	-
PATCH	<code>/:id</code>	Update round	✓	Organizer (owner)
DELETE	<code>/:id</code>	Delete round	✓	Organizer (owner)

Criteria Routes (`/api/criteria`)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/create/:roundId</code>	Create criteria for round	✓	Organizer (owner)
GET	<code>/round/:roundId</code>	Get all criteria for round	✓	-
GET	<code>/:id</code>	Get criteria by ID	✓	-
PATCH	<code>/:id</code>	Update criteria	✓	Organizer (owner)
DELETE	<code>/:id</code>	Delete criteria	✓	Organizer (owner)

Team Routes (`/api/teams`)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/create/:hackathonId</code>	Create team	✓	Participant
GET	<code>/hackathon/:hackathonId</code>	Get all teams for hackathon	✓	-
GET	<code>/:id</code>	Get team by ID	✓	-
PATCH	<code>/:id</code>	Update team details	✓	Team Leader
DELETE	<code>/:id</code>	Delete team	✓	Team Leader
POST	<code>/:id/add-member</code>	Add member to team	✓	Team Leader
POST	<code>/:id/remove-member</code>	Remove member from team	✓	Team Leader
POST	<code>/:id/transfer-leadership</code>	Transfer team leadership	✓	Team Leader
POST	<code>/:id/leave</code>	Leave team	✓	Team Member

Submission Routes (`/api/submissions`)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/create/:roundId</code>	Submit project for round	✓	Team Leader
GET	<code>/round/:roundId</code>	Get all submissions for round	✓	Judge/Organizer
GET	<code>/team/:teamId</code>	Get all submissions by team	✓	Team Member
GET	<code>/:id</code>	Get submission by ID	✓	-
PATCH	<code>/:id</code>	Update submission	✓	Team Leader
DELETE	<code>/:id</code>	Delete submission	✓	Team Leader/Admin

★ Evaluation Routes (`/api/evaluations`)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/create/:submissionId</code>	Evaluate submission	✓	Judge
GET	<code>/submission/:submissionId</code>	Get evaluations for submission	✓	Judge/Organizer
GET	<code>/round/:roundId</code>	Get all evaluations for round	✓	Organizer
GET	<code>/:id</code>	Get evaluation by ID	✓	Judge (owner)
PATCH	<code>/:id</code>	Update evaluation	✓	Judge (owner)
DELETE	<code>/:id</code>	Delete evaluation	✓	Judge (owner)/Admin
GET	<code>/judge/my-evaluations</code>	Get judge's evaluations	✓	Judge










Result Routes (`/api/results`)

Method	Endpoint	Description	Auth Required	Role Required
POST	<code>/calculate/round/:roundId</code>	Calculate round results	✓	Organizer (owner)
POST	<code>/calculate/hackathon/:hackathonId</code>	Calculate overall results	✓	Organizer (owner)
GET	<code>/round/:roundId</code>	Get round results	✓	-
GET	<code>/hackathon/:hackathonId</code>	Get overall hackathon results	✓	-
GET	<code>/team/:teamId</code>	Get team results	✓	-
POST	<code>/publish/round/:roundId</code>	Publish round results	✓	Organizer (owner)
POST	<code>/publish/hackathon/:hackathonId</code>	Publish overall results	✓	Organizer (owner)

Summary




Total Endpoints: 67

Breakdown by Module:

-  Authentication: 6 endpoints
 -  Users: 7 endpoints
 -  Hackathons: 9 endpoints
 -  Rounds: 5 endpoints
 -  Criteria: 5 endpoints
 -  Teams: 9 endpoints
 -  Submissions: 6 endpoints
 -  Evaluations: 7 endpoints
 -  Results: 7 endpoints
-

Build Order (Recommended)

Already Built:

1. Auth Controller 
2. User Controller 
3. Hackathon Controller 

Build Next (in order):

Phase 1: Core Features 4. **Round Controller** (depends on Hackathon) 5. **Criteria Controller** (depends on Round) 6. **Team Controller** (depends on Hackathon)

Phase 2: Submission & Evaluation 7. **Submission Controller** (depends on Round & Team) 8. **Evaluation Controller** (depends on Submission)

Phase 3: Results 9. **Result Controller** (depends on all above)

Key Points to Remember:

Authorization Checks:

- **Hackathon Owner:** Check `hackathon.organizer === req.user._id`

- **Team Leader:** Check `team.leader === req.user._id`
- **Team Member:** Check if user is in `team.members` or is leader
- **Judge for Hackathon:** Check if user is in `hackathon.judges`

Common Validations:

- Check if resource exists before updating/deleting
- Check user authorization before allowing action
- Trim string inputs
- Validate required fields
- Check business logic (e.g., can't submit after deadline)

Populate References:

- Use `.populate()` to get related data
- Example: `.populate("organizer", "name email")`
- Example: `.populate({ path: "teams", populate: { path: "leader members" } })`

Error Handling:

- Use `throw new apiError(statusCode, message)`
 - Always wrap in `asyncHandler`
 - Return proper HTTP status codes
-

Need Help?

When building, if you get stuck:

1. **Check the pattern** in Auth/User/Hackathon controllers
2. **Ask specific questions** like:
 - "How do I check if user is team leader?"
 - "How do I populate nested references?"
 - "What's the best way to validate this?"
3. **Share your code** and I'll review/debug
4. **Test each endpoint** as you build it

Good luck! You've got this! 💪