Freelink API Documentation

Freelink Development Team

Version 1.0.0

A comprehensive API for a freelancing platform system

Contents

1 Authentication

1.1 POST /api/users/login/

Authenticate user and return authentication token with user details.
Request Body:

- `email` (string): User's email address
- `password` (string): User's password

Response:

- `200 OK`: Returns authentication token and user details

1.2 POST /api/users/register/

Register a new user account.
Request Body:

- `email` (string): User's email address
- `password` (string): User's password
- `first_name` (string): User's first name
- `last_name` (string): User's last name
- `user_type` (string): Either "client" or "freelancer"

Response:

- `201 Created`: User account created successfully

1.3 POST /api/users/logout/

Log out user by deleting token and ending session.
Response:

- `200 OK`: No response body

1.4 POST /api/users/password-reset-request/

Request a password reset link by providing email address.
Request Body:

- `email` (string): User's email address

Response:

- `200 OK`: Password reset email sent

1.5 POST /api/users/reset-password/

Reset password for unauthenticated users using UID and token.
Request Body:

- `uid` (string): User ID
- `token` (string): Reset token
- `new_password` (string): New password

Response:

- `200 OK`: Password reset successful

1.6 PUT /api/users/change-password/

Change password for the authenticated user.
Request Body:

- `current_password` (string): Current password
- `new_password` (string): New password

Response:

- `200 OK`: Password changed successfully

2 Users

2.1 POST /api/users/verify-email/

Verify email address for a newly registered user.
Request Body:

- `uid` (string): User ID
- `token` (string): Verification token

Response:

- `200 OK`: Email verified successfully

3 Profiles

3.1 GET /api/profiles/me/

View your own profile.
Response:

- `200 OK`: Returns user's profile data

3.2 GET /api/profiles/{user___email}/

View a public profile by email.
Parameters:

- `user__email` (path, string, required): User's email address

Response:

- `200 OK`: Returns public profile data

3.3 PUT /api/profiles/me/update/

Update your own profile.
Request Body:

- `bio` (string): Personal biography
- `skills` (array): Array of skills
- `portfolio_url` (string): Portfolio website URL

- `hourly_rate` (number): Hourly rate for freelancers
- `availability` (string): Availability status

Response:

- `200 OK`: Profile updated successfully

4 Jobs

4.1 GET /api/jobs/

List all available jobs.
Response:

- `200 OK`: Returns list of jobs

4.2 POST /api/jobs/

Create a new job (clients only).
Request Body:

- `title` (string): Job title
- `description` (string): Job description
- `budget` (number): Project budget
- `skills_required` (array): Required skills
- `deadline` (string): Project deadline (ISO format)

Response:

- `201 Created`: Job created successfully

4.3 GET /api/jobs/{id}/

Retrieve a specific job by ID.
Parameters:

- `id` (path, integer, required): Job ID

Response:

- `200 OK`: Returns job details

4.4 PUT /api/jobs/{id}/

Update a job (only the client who created it).
Parameters:

- `id` (path, integer, required): Job ID

Request Body: Same as POST /api/jobs/
Response:

- `200 OK`: Job updated successfully

4.5 DELETE /api/jobs/{id}/

Delete a job (only the client who created it).
Parameters:

- `id` (path, integer, required): Job ID

Response:

- `204 No Content`: No response body

4.6 PUT /api/jobs/{id}/status/

Update job status (e.g., mark as completed or cancelled).
Parameters:

- `id` (path, integer, required): Job ID

Request Body:

- `status` (string): New status (e.g., "completed", "cancelled")

Response:

- `200 OK`: Status updated successfully

5 Skills Management

5.1 GET /api/jobs/skills/

List all available skills.
Response:

- `200 OK`: Returns list of skills

5.2 POST /api/jobs/skills/

Create a new skill (admin only).
Request Body:

- `name` (string): Skill name
- `category` (string): Skill category

Response:

- `201 Created`: Skill created successfully

5.3 DELETE /api/jobs/skills/{id}/

Remove a skill (admin only).
Parameters:

- `id` (path, integer, required): Skill ID

Response:

- `204 No Content`: No response body

6 Proposals

6.1 GET /api/proposals/

List proposals:

- Freelancers: Their own proposals
- Clients: Proposals received for their jobs

Response:

- `200 OK`: Returns list of proposals

6.2 POST /api/proposals/

Submit a new proposal for a job (freelancers only).
Request Body:

- `job_id` (integer): Job ID
- `cover_letter` (string): Proposal cover letter
- `bid_amount` (number): Proposed amount
- `timeline` (string): Proposed timeline

Response:

- `201 Created`: Proposal submitted successfully

6.3 GET /api/proposals/{id}/

Retrieve a single proposal.
Parameters:

- `id` (path, integer, required): Proposal ID

Response:

- `200 OK`: Returns proposal details

6.4 PUT /api/proposals/{id}/status/

Update proposal status (clients only).
Parameters:

- `id` (path, integer, required): Proposal ID

Request Body:

- `status` (string): "accepted" or "declined"

Actions:

- **Accept**: Creates a contract, marks job as 'in_progress', declines other proposals
- **Decline**: Marks proposal as declined

Response:

- `200 OK`: Status updated successfully

7 Contracts

7.1 GET /api/contracts/contracts/

List all contracts for the authenticated user.
Response:

- `200 OK`: Returns list of contracts

7.2 POST /api/contracts/contracts/

Create a new contract (clients only, via proposal acceptance).
Request Body:

- `proposal_id` (integer): Proposal ID to accept
- `terms` (string): Contract terms and conditions
- `milestones` (array): Array of milestone objects

Response:

- `201 Created`: Contract created successfully

7.3 GET /api/contracts/contracts/{id}/

Retrieve contract details.
Parameters:

- `id` (path, string, required): Contract UUID

Response:

- `200 OK`: Returns contract details

7.4 PATCH /api/contracts/contracts/{id}/accept/

Freelancer accepts a contract in 'pending_acceptance' state.
Parameters:

- `id` (path, string, required): Contract UUID

Response:

- `200 OK`: Contract accepted

7.5 PATCH /api/contracts/contracts/{id}/reject/

Freelancer rejects a contract in 'pending_acceptance' state.
Parameters:

- `id` (path, string, required): Contract UUID

Response:

- `200 OK`: Contract rejected

7.6 PATCH /api/contracts/contracts/{id}/submit-work/

Freelancer submits work (moves contract from 'active' to 'in_review').
Parameters:

- `id` (path, string, required): Contract UUID

Response:

- `200 OK`: Work submitted for review

7.7 PATCH /api/contracts/contracts/{id}/dispute/

Raise a dispute on an active or in-review contract.
Parameters:

- `id` (path, string, required): Contract UUID

Response:

- `200 OK`: Dispute raised

7.8 GET /api/contracts/contracts/user/{user_id}/

List all contracts for a specific user.
Parameters:

- `user_id` (path, integer, required): User ID

Response:

- `200 OK`: Returns user's contracts

8 Milestones

8.1 GET /api/contracts/contracts/{contract_id}/milestones/

List milestones for a contract.
Parameters:

- `contract_id` (path, string, required): Contract UUID

Response:

- `200 OK`: Returns list of milestones

8.2 POST /api/contracts/contracts/{contract_id}/milestones/

Add a milestone to a contract (only if contract is active).
Parameters:

- `contract_id` (path, string, required): Contract UUID

Request Body:

- `title` (string): Milestone title
- `description` (string): Milestone description
- `amount` (number): Milestone value
- `due_date` (string): Due date (ISO format)

Response:

- `201 Created`: Milestone added

8.3 PUT /api/contracts/contracts/{contract_id}/milestones/{id}/

Update a milestone.
Parameters:

- `contract_id` (path, string, required): Contract UUID
- `id` (path, string, required): Milestone UUID

Request Body: Same as POST
Response:

- `200 OK`: Milestone updated

8.4 DELETE /api/contracts/contracts/{contract_id}/milestones/{id}/

Remove a milestone.
Parameters:

- `contract_id` (path, string, required): Contract UUID
- `id` (path, string, required): Milestone UUID

Response:

- `204 No Content`: No response body

9 Messaging

9.1 GET /api/chat/inbox/

List all messages received by the logged-in user.
Response:

- `200 OK`: Returns list of received messages

9.2 GET /api/chat/sent/

List all messages sent by the logged-in user.
Response:

- `200 OK`: Returns list of sent messages

9.3 GET /api/chat/message/{username}/

List messages between logged-in user and specified username. Marks messages as read if recipient is current viewer.
Parameters:

- `username` (path, string, required): Other user's username

Response:

- `200 OK`: Returns conversation history

9.4 POST /api/chat/send/

Send a new message.
Request Body:

- `recipient` (string): Recipient's username
- `message` (string): Message content

Response:

- `201 Created`: Message sent successfully

10 Payments & Wallet

10.1 GET /api/wallet/wallet/

View wallet balance and transaction history.
Response:

- `200 OK`: Returns wallet information

10.2 POST /api/payments/init/

Initialize a payment/deposit.
Request Body:

- `amount` (number): Deposit amount
- `currency` (string): Currency code (e.g., "USD")

Response:

- `200 OK`: Returns payment initialization data

10.3 GET /api/payments/verify/

Verify a payment transaction.
Response:

- `200 OK`: Returns payment verification status

10.4 GET /api/escrow/escrow/{transaction_id}/

View escrow transaction details.
Parameters:

- `transaction_id` (path, integer, required): Transaction ID

Response:

- `200 OK`: Returns escrow details

11 Ratings

11.1 GET /api/ratings/{user_id}/

View ratings for a specific user.
Parameters:

- `user_id` (path, integer, required): User ID

Response:

- `200 OK`: Returns user ratings

11.2 POST /api/ratings/{user_id}/

Submit a rating for a user.
Parameters:

- `user_id` (path, integer, required): User ID to rate

Request Body:

- `rating` (integer): Rating value (1-5)
- `comment` (string): Review comment
- `contract_id` (string): Related contract UUID

Response:

- `201 Created`: Rating submitted successfully

12 Notifications

12.1 GET /api/notifications/

Get user notifications.
Response:

- `200 OK`: Returns list of notifications

13 Dashboard

13.1 GET /api/dashboard/

Get dashboard overview data.
Response:

- `200 OK`: Returns dashboard statistics and overview

14 Transactions

14.1 GET /api/transactions/

View transaction history.
Response:

- `200 OK`: Returns transaction history

14.2 PUT /api/transactions/{transaction_id}/

Update transaction status.
Parameters:

- `transaction_id` (path, integer, required): Transaction ID

Request Body:

- `status` (string): New status

Response:

- `200 OK`: Status updated successfully

15 Notes

- All endpoints require authentication unless specified otherwise.
- UUID parameters refer to universally unique identifiers.
- HTTP status codes follow REST conventions.
- Error responses include appropriate status codes and error messages.

For any questions or support, please contact the development team.