# Formatting

Each section is ordered GET 🡪 POST 🡪 PUT 🡪 DELETE.

Highlighted sections do not have a unique request and include a description of which request provides the functionality of the highlighted request.

**<bracketedBoldedFields>** represent fields which should be replaced by their corresponding value

# Account Management

## View Profile:

### GET /accounts/**<accountId>**?

*Returns account’s first name, last name. If account is not private, returns bio, bio link, following, followers, and rating.*

loggedInId=**<loggedInId>** Current account id of logged in user

## Get Bio:

### *See “View Profile” (GET /accounts/****<accountId>****)*

## Get Privacy:

### *See “View Profile” (GET /accounts/****<accountId>****)*

## Get Pinned:

### *See “View Profile” (GET /accounts/****<accountId>****)*

## Create Account:

### POST /accounts?

Creates a new account with the specified information

firstName=**<firstName>**& User’s first name

lastName=**<lastName>**& User’s last name

email=**<email>**& User’s email

password=**<password>** User’s password

## Login:

### POST /accounts/login?

Returns accountId if successful, -1 if unsuccessful

email=**<email>**& User’s email

password=**<password>** User’s password

## Update Bio:

### PUT /accounts/**<accountId>**/bio?

Updates account’s bio based on given information

bio=**<bio>**& Optional, new bio for user

bioLink=**<bioLink>** Optional, new bio link for user

## Update Privacy:

### PUT /accounts/**<accountId>**/private?

Sets account privacy to given value (true or false)

private=**<private>** True for private, False for public

## Pin Post:

### PUT /accounts/**<accountId>**/pinned?

Pins the given post ID for the given user

pinnedPostId=**<postId>** Optional, new post id to pin

# Posting and Commenting

## Get Posts with Given Filtering:

### GET /posts?

*Returns all posts with the given filtering fields, all of which are optional.*

*Note that checking substrings may be much more difficult server-side than client-side, which may lead to any string-related filtering being done client-side.*

products=**<products>**& Contains given products (using substring)

instructions=**<instructions>**& Contains given instructions (using substring)

lookKind=**<lookKind>**& Contains given look (using substring)

minLookTime=**<minTime>**& Takes at least this long (inclusive)

maxLookTime=**<maxTime>**& Takes no longer than this long (inclusive)

minDifficulty=**<minDiff>**& At least this difficult (inclusive)

maxDifficulty=**<maxDiff>**& No harder than this (inclusive)

sortBy=**<sortType>** Keywords:

new OR old, fast OR slow, easy OR hard

Sorts by given keywords, in order of keywords

Competing keywords (ex. fastslow) cancel out

## Get Posts from a Specific Author:

### GET /posts/authors/**<authorId>**

*If the author is public, returns all the posts, reposts, and comments from the specified author, ordered from newest to oldest. If the author is private, returns an error message. If the author does not exist, returns a different error message.*

## Get Post Reviews on a Specific Post:

### GET /posts/posts/**<postId>**

*Returns all the posts reviewing the specified post, should be used in combination with “Get Comments on a Specific Post”*

## Get Comments from a Specific Author:

### *See “Get Posts from a Specific Author” (GET /posts/authors/****<authorId>****)*

## Get Comments on a Specific Post:

### GET /comments/posts/**<commentId>**

*Returns all the comments on the specified post, should be used in combination with “Get Post Reviews on a Specific Post”*

## Get Comments on a Specific Comment:

### GET /comments/comments/**<commentId>**

*Returns all the comments on the specified comment, there is no corresponding “Get Posts on a Specific Comment” because you cannot reply to a comment with a post.*

## Publish Look:

### POST /posts/post?

Creates a post with the given information

authorId=**<authorId>**& Poster’s user id

parentPostId=**<postId>**& Optional, if post is a review of a previous post

photo=**<photo>**& Photo converted to a Blob

products=**<products>**& String of products used

instructions=**<instructions>**& String of instructions for look

lookKind=**<lookKind>**& String describing kind of look

lookTime=**<lookTime>**& Int of time (minutes) for look

lookDifficulty=**<difficulty>** Int of look difficulty on scale 1-10

## Publish Comment:

### POST /comments/comment?

*Creates a comment or repost with the given information*

authorId=**<authorId>**& Poster’s user id

parentPostId=**<id>**& Used for replies to posts

parentCommentId=**<id>**& Used for replies to comments

Exactly one of parentPostId and parentCommentId

must be specified

comment=**<comment>**& Poster’s comment, optional for a repost

isRepost=**<isRepost>**& Boolean whether the comment is a repost

restricted=**<restricted>** Boolean whether the repost is restricted

Only used for reposts, default false

# Ratings, Reactions, and Followers

## Get Followers/Following:

### See “View Profile” (GET /accounts/**<accountId>**)

## Get Ratings for an Author:

### GET /ratings/**<authorId>**

*Returns all the ratings for a given author*

## Get Reactions for a Post:

### GET /reactions/posts/**<postId>**

*Returns all the reactions for a given post*

## Get Reactions for a Comment:

### GET /reactions/comments/**<commentId>**

Returns all the reactions for a given comment

## Publish Rating:

### POST /ratings/rating?

Creates a rating by one user about another

authorId=**<authorId>**& The id of the user getting rated

raterId=**<raterId>**& The id of the rating user

score=**<score>**& Score on a scale 1-10 as the rating

comment=**<comment>** Optional comment to attach to a rating

## Publish Reaction:

### POST /reactions/reaction?

*Creates a reaction (like or dislike) by one user about a post*

parentPostId=**<postId>**& The id of the post being reacted to

reactorId=**<reactorId>**& The id of the user reacting to the post

isPositive=**<isPositive>** Boolean, true means like, false means dislike

## Follow/Unfollow:

### POST /followers/follow?

Allows one user to follow another

leaderId=**<leaderId>**& The id of the user being followed

followerId=**<followerId>**& The id of the user following

### DELETE /followers/follow?

Allows one user to stop following another

leaderId=**<leaderId>**& The id of the user being unfollowed

followerId=**<followerId>**& The id of the user unfollowing