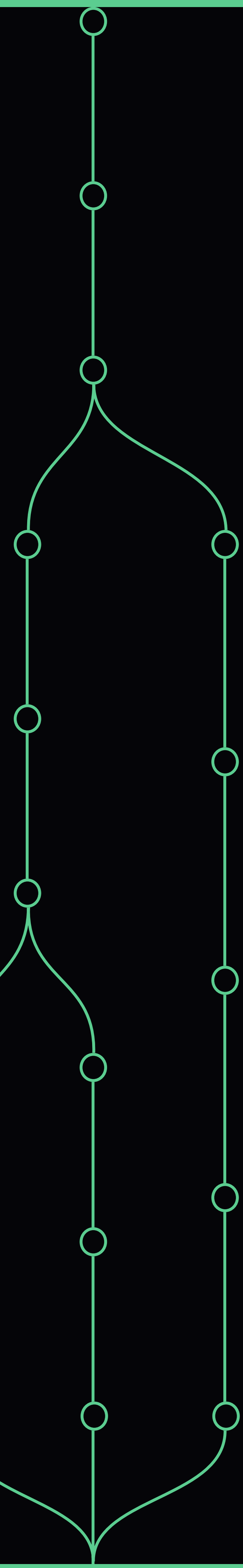
Usof backend

Track Full Stack

September 1, 2025



Innovation Campus

ucode connect

Contents

Engage

DESCRIPTION

Greetings to you!

This is your first challenge in the FullStack Track. It's time to show what you are capable of.

As people's skill sets get increasingly specialized, the practice of collaboration becomes more important than ever. Collaboration is when a group of people come together and contribute their expertise for the benefit of a shared objective, project or mission. It's a learned skill. The outcome of the solution greatly depends on how well you can collaborate with others. When a group of people combines their knowledge, skills, experience, and discusses problems and their possible solutions, then the development process will certainly keep moving forward.

One of the best things about working with people who have different skills and backgrounds is experience exchange. This is the idea behind StackOverflow, a popular question and answer system (Q&A) about programming. The objective is to increase work effectiveness by sharing experience, insights and knowledge.

To operate with various data on this resource you need some kind of interface on the site's backend. Moreover, it must be compatible with client apps on different platforms.

This can be achieved using an application programming interface (API). An API is a set of functions and procedures that allow to create apps for accessing data and functions of other apps.

You have already worked with APIs during the Half Marathon. Now, it's time to build one!

BIG IDEA

Knowledge and experience exchange.

ESSENTIAL QUESTION

How to help people exchange knowledge?

CHALLENGE

Develop an API as the first step in creating your website.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find solutions, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- How to simplify the procedure of counseling?
- Which of the skills that you acquired during the Half Marathon could be useful in this challenge?
- What resources of knowledge sharing do you know?
- What are the disadvantages of APIs?
- Which advantage of APIs is also a disadvantage?
- What do you think is more useful - a knowledge sharing website or official documentation?
- What makes StackOverflow so popular?
- What functionality will make your website better? How can it be improved?
- How can you make a product unique and useful?
- What profit can you get from usof ?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Investigate what features are critically important in every website.
- Ask students who have already started this challenge which information you need to find first of all.
- Investigate the main features and functionality of databases. Choose the one you want to implement for the future website. Be ready to explain your choice.
- Find what role the admin panel plays in websites and whether it is necessary.
- Research why APIs are useful for researchers and developers.
- Use Miro or Trello to break the challenge into pieces and visualize it.
- Clone your git repository that is issued on the challenge page.
- Start to develop your solution.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.
- Perform only those tasks that are given in the story.
- Analyze all information you have collected during the preparation stages.
- Complete the Document section while developing a challenge. It is described after the Act section.
- You can proceed to Act: Creative only after you have completed all requirements in Act: Basic. But before you begin to complete the challenge, pay attention to the program's architecture. Take into account the fact that many features indicated in the Act: Creative require special architecture. And in order not to rewrite all the code somewhen, we recommend you initially determine what exactly you will do in the future. Note that the Act: Basic part gives the minimum points to validate the challenge.
- You can use the open-source admin panel. For example, take a look at it
- Validate all requests with relevant responses.
- Be sure to properly organize access to data via endpoints based on users' roles.
- Complete tasks according to the rules specified in the following style guides:
    - JavaScript:
        - JavaScript Style Guide and Coding Conventions
        - JavaScript Best Practices
        - Node.js Best Practices
        - SQL style guide
- The solution will be checked and graded by students like you.
  Peer-to-Peer learning.
- If you have any questions or don't understand something, ask other students or just Google it.

Act: Basic

ALLOWED STACK

`JS, Node.js, Express, MySQL`

DESCRIPTION

Your challenge now is to create an API for a future question and answer service for professional and enthusiast programmers - usof.
The latter allows you to share your problems/solutions with short posts and receive solutions/feedback, or even increase your profile rating. You have probably guessed what it looks like.

Your API must allow users to register, view posts, like and comment on them. This challenge is only the first part of your future big website. Keep it in mind.

Develop the server and database for your app in this challenge. Create the API server. That means that you have to create the API responses for a web app. For this use Express - Node.js web application framework.
You must use a relational Mysql database.

Add an admin panel to manipulate data (posts, categories, users, etc.) which is accessible only for users with admin privileges.

Your database must be created (and recreated, if necessary) upon initialization of the solution. Save all files, such as user photos, in your server local files system (for example, specified directory on your PC). Fill the database with some test data (at least five entries per each table, more will be better).

Do not forget about error handling. It must be informative and useful.

Create your app with an MVC pattern.
Your code must be written with the OOP paradigm and SOLID principles.

FUNCTIONALITY

At first, create an admin panel to manipulate data (for example, posts, categories, users, etc.) only for users with admin access rights. You must implement user authorization by yourself.

The admin must have an opportunity to make all CRUD operations with entities.

In the architecture of your backend, implement at least two kinds of roles: user and admin. Their functionality will be described below

Your API must allow to implement the following functionality for admins:

- authorization :
  - log in
  - log out
  - reset password
- user :
  - create user/admin
  - see all profiles
  - update profile data
  - delete
- post :
  - create
  - see all, even inactive
  - update data: change category or set any post active/inactive if its content is unacceptable for users or the problem has been resolved. Content is not editable even for admins. Inactive needs to be hidden from users
  - delete
- category :
  - create
  - see all
  - update
  - delete
- comment :
  - create
  - see all
  - change status to active/inactive. Content is not editable even for admins
  - delete
- like :
  - create only one (like/dislike) under any post or comment - associated with the same admin profile
  - see all likes on posts or comments
  - delete

Your API must allow to implement the following functionality for users:

- authorization :
  - register - everyone can register on the website
  - log in
  - log out
  - reset password
- post :
  - create
  - see only active posts of all users and personal inactive posts
  - update data at self created posts - change category and content
  - delete
- comment :
  - create comments under active posts
  - see all comments for the specified post
  - update any - only change status to active/inactive
  - delete
- like :
  - like - create only one (like/dislike) under any post or comment
  - see all likes under the specified active post or comment
  - delete self-created likes

You are welcome to implement additional functionality if you think it will be useful. Be ready to explain your choice in case your assessor asks for it.

ENTITIES

Every entity must contain the fields listed below:

- User entity:
  - login - must be unique
  - password - use best practices for password storage
  - full name - real user name, some validation of this data is recommended
  - email - user personal email, you have to check whether it is real and belongs to the user. A unique link to email is a good idea, but you can choose another way
  - profile picture - photo of the user or some picture which will be associated with the user's profile
  - rating - the sum of all likes under all posts and comments. You have to subtract dislikes from this value, of course. It is calculated automatically for each user
  - role - admin/user, all users have user role by default and only an admin can change it. You can add other roles (for example, guest, etc.) if you see a reason for it

- Post entity:
  - author - the user who has created the post
  - title
  - publish date
  - status - active or inactive
  - content - text for problem explanation, opportunity to add images is highly recommended
  - categories - associated categories
    Note that there can be several categories for one post.
- Category entity:
  - title
  - description
- Comment entity:
  - author
  - publish date
  - content
- Like entity:
  - author
  - publish date
  - post/comment id - under what entity the like is added
  - type - like or dislike

You are allowed to create additional fields and even entities if you need to. Be ready to explain your choice in case your assessor asks for it.

ENDPOINTS

Below is a possible endpoint structure of your service. You may choose a different structure, but be prepared to explain your choices.

- Authentication module:
  - POST - /api/auth/register - registration of a new user, required parameters are [login, password, password confirmation, email]
  - POST - /api/auth/login - log in user, required parameters are [login, email, password]. Only users with a confirmed email can sign in
  - POST - /api/auth/logout - log out authorized user
  - POST - /api/auth/password-reset - send a reset link to user email, required parameter is [email]
  - POST - /api/auth/password-reset/:confirm_token - confirm new password with a token from email, required parameter is a [new password]
- User module:
  - GET - /api/users - get all users
  - GET - /api/users/:user_id - get specified user data
  - POST - /api/users - create a new user, required parameters are [login, password, password confirmation, email, role]. This feature must be accessible only for admins
  - PATCH - /api/users/avatar - upload user avatar
  - PATCH - /api/users/:user_id - update user data
  - DELETE - /api/users/:user_id - delete user

- Post module:
  - GET - /api/posts - get all posts. This endpoint doesn't require any role, it is public. If there are too many posts, you must implement pagination. Page size is up to you
  - GET - /api/posts/:post_id - get specified post data. Endpoint is public
  - GET - /api/posts/:post_id/comments - get all comments for the specified post. Endpoint is public
  - POST - /api/posts/:post_id/comments - create a new comment, required parameter is [content]
  - GET - /api/posts/:post_id/categories - get all categories associated with the specified post
  - GET - /api/posts/:post_id/like - get all likes under the specified post
  - POST - /api/posts/ - create a new post, required parameters are [title, content, categories]
  - POST - /api/posts/:post_id/like - create a new like under a post
  - PATCH - /api/posts/:post_id - update the specified post (its title, body or category). It's accessible only for the creator of the post
  - DELETE - /api/posts/:post_id - delete a post
  - DELETE - /api/posts/:post_id/like - delete a like under a post
- Categories module:
  - GET - /api/categories - get all categories
  - GET - /api/categories/:category_id - get specified category data
  - GET - /api/categories/:category_id/posts - get all posts associated with the specified category
  - POST - /api/categories - create a new category, required parameter is [title]
  - PATCH - /api/categories/:category_id - update specified category data
  - DELETE - /api/categories/:category_id - delete a category
- Comments module:
  - GET - /api/comments/:comment_id - get specified comment data
  - GET - /api/comments/:comment_id/like - get all likes under the specified comment
  - POST - /api/comments/:comment_id/like - create a new like under a comment
  - PATCH - /api/comments/:comment_id - update specified comment data
  - DELETE - /api/comments/:comment_id - delete a comment
  - DELETE - /api/comments/:comment_id/like - delete a like under a comment

Attention
Don't forget about a feature that allows locking posts/comments. Think about how you will implement it.

You can create additional endpoints if you need to. Be ready to explain your choices in case your assessor asks.

You also must implement post sorting and filtering.
Every user must have the opportunity to sort all viewable posts:

- by number of likes - by default
- by date

The user must have the opportunity to filter all viewable posts as well:

- by categories
- by date interval
- by status

You can add additional sorting and filtering conditions if you need it.

Act: Creative

DESCRIPTION

It is the place where your imagination and creativity play a significant role. Implement additional features to make the program better and more unique. Listed below are a few ideas you can add to your program. You can come up with everything you want to improve your program. Creative features:

- implement an opportunity to add posts to the Favorites category, so that the user can quickly view all marked posts by going on the appropriate endpoint
- allow users to subscribe to some posts that they want to follow and send them notifications when some kind of activity was made with the post (it was changed somehow or it was commented by somebody)
- add more functionality to your API or make it more useful and user friendly
- other creative features

Document

DOCUMENTATION

One of the attributes of Challenge Based Learning is documentation of the learning experience from challenge to solution. Throughout the challenge, you document your work using text and images and reflect on the process. These artifacts are helpful for ongoing reflection, informative assessment, evidence of learning, portfolios, and telling the story of challenge. The end of each phase (Engage, Investigate, Act) of the challenge offers an opportunity to document the process.

Much of the most profound learning occurs by considering the process, thinking about one's own learning, analyzing ongoing relationships with the content and between concepts, interacting with other people, and developing a solution. During learning, documentation of all processes will help you analyze your work, approaches, thoughts, implementation options, code, etc. In the future, this will help you understand your mistakes, improve your work, and read the code.

It is vital to understand and do this at the learning stage, as this is one of the skills you will need in your future job. Naturally, the documentation should not be voluminous. It should be presented in an accessible, logical, and connected form.

So what should you do?

- A nice-looking and helpful README file. For people to want to use your product, their first introduction must be through the README on the project's git page. Your README file must contain:
  - Short description. It means that there must be some info about what your project is. For example, what your program does.
  - Screenshots of your solution. This point is about screenshots of your project "in use".
  - Requirements and dependencies. A list of everything that needs to be installed on any machine to build your project.
  - How to run your solution. Describe the steps from cloning your repository to the first launch of your program.
- Full-fledged documentation in any form is convenient for you. By writing this, you will get some benefits:
  - you have an opportunity to think through implementation without the overhead of changing code every time you change your mind about how something should be organized. You will have excellent documentation available for you to know what you need to implement
  - if you work with a development team and they have access to this information before you complete the project, they can confidently start working on another part of projects that will interact with your code
  - everyone can find how your project works
- Your documentation must contain:
  - description of progress after every competed CBL stage
  - description of the algorithm of your whole program

Keep in mind that peers will check the implementation of this stage at the assessment!

Also, several links can help you:

- Make a README
- How to write a readme.md file?
- A Beginners Guide to writing a README
- Google Tools - a good way to journal your phases and processes:
  - Google Docs
  - Google Sheets
- Git Wiki - a section for hosting documentation on Git-repository
- Haroopad - a markdown enabled document processor for creating web-friendly documents
- Canva - a good way to visualize your data
- code commenting - source code clarification method. The programming language determines the syntax of comments
- and others to your taste

Share

DESCRIPTION

Last but not least, the final stage of your work is to publish it on LinkedIn in a form of post. This step isn't just about showcasing your work — it's a crucial part of Challenge Based Learning framework. During this stage, you will discover ways of getting external evaluation and feedback on your work. Analyzing your process helps you to understand your strengths and areas for improvement, while sharing your insights invites valuable feedback from peers and professionals.

Pay attention to details:

- aim to make your post both informative and reflective
- start by briefly introducing the challenge and its purpose
- highlight the key challenges you faced and the technologies or solutions you used to overcome them
- share any significant findings or lessons learned during the process
- conclude with a reflection on how this experience has prepared you for future work and invite others to share their thoughts

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime or OBS - an easy way to capture your screen, record video or audio

Don’t forget to tag your post with #InnovationCampusKhPI to connect with the community! Follow and mention via @ Innovation Campus of NTU "KhPI"!