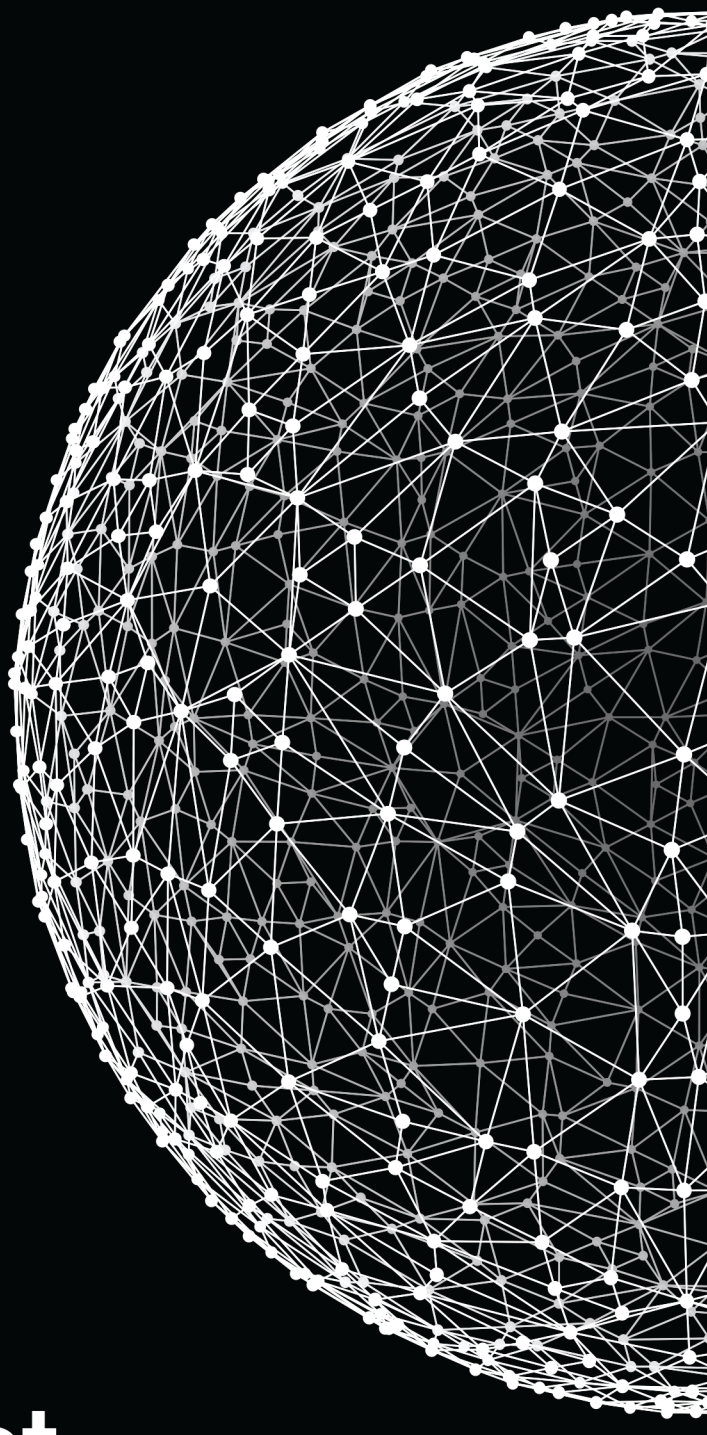
uchat

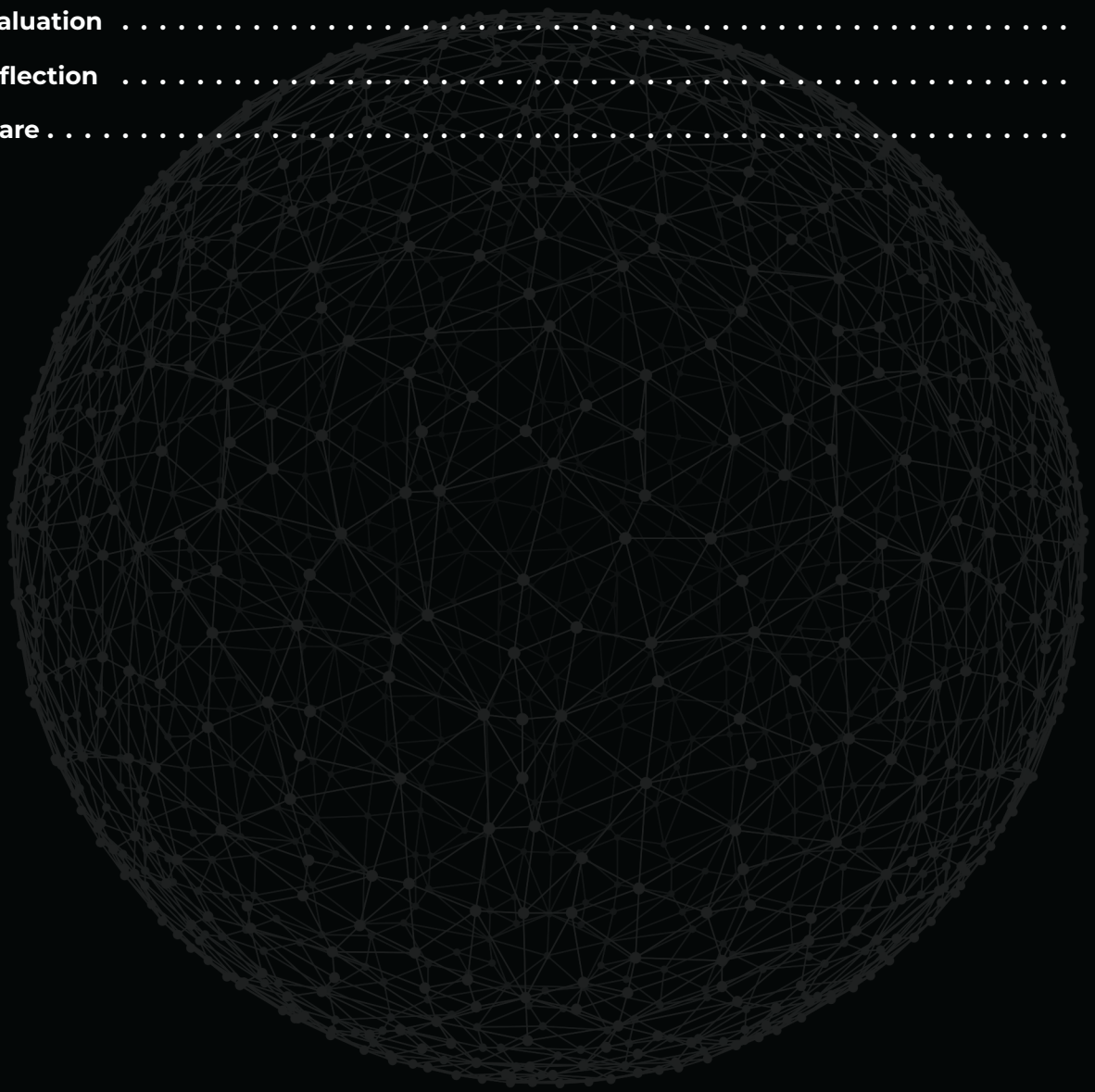
Track C

December 24, 2020



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Here we go again!
Track C is coming to a close. What does it mean?
You're ready for the last big product challenge in C. As it was during Marathon.

What does the product challenge mean?
You gather a team, research a field, discover a real-world problem and develop your solution that solves its tasks. You are creating your own product.
As always: each step of Challenge Based Learning framework is required. No grit, spit and a whole lot of duct tape approach. Take your time! Stop at each stage, research it, make sure that all team members understand what is at stake. Get the most out of your teamwork.
This challenge is essential for your education and future career. We expect that you've gained the Basics of Computer Science and will use all your consolidated knowledge during this challenge to get the best product you've ever made.

BIG IDEA

Product building.

ESSENTIAL QUESTION

How to create own product that solves a real-world problem with limited resources using C?

CHALLENGE

Messaging apps have become an essential tool in everyday life and an undisputed leader on all app markets. At the same time, the concept and definition of such apps is forever changing, with new features and ideas being introduced every day. The market is very competitive, making innovations highly sought-after.

We invite you to accept the battle and create your own messaging app in C. Right product at the right time.

This challenge allows you to figure out what makes messaging apps popular. You will need to find out why users choose certain apps to others, what features they are interested in, and do something that will win the attention of your friends, colleagues, and, finally, the whole market.

Investigate

GUIDING QUESTIONS

Gather your team in one place. You can book a meeting room, sit in the cantina, or just gather in a cluster. Discuss the guiding questions and any further questions that will arise during your discussion. This will help each team member to be aware of what you need to do. Note down all discussed answers and ideas.

You already have a bit of experience in creating products, so you know that just pouncing on a task without preparing is at least unproductive and ineffective. But it is very convenient to divide any task into topics of which it consists.

One of these topics is the architecture of the future project. Client-server model was a breakthrough in its time. Many client programs can access the server. There is no need to copy the server's code on every machine. It is not even necessary to have enormous processing power on clients' machines. Furthermore, there is a convenient security adjustment. What else is needed?

Before moving on, we encourage you to find answers to the following questions to get a better understanding of the topic.

- What are the most common layers in multitier architecture?
- What is the difference between client and server?
- What is a request-response method?
- What are the pull and push technologies?
- What is the limit number of clients?
- How many servers do we need to run a simple client-server model?
- Which popular apps use the client-server model?
- What are the tools for transferring data between client and server?
- What is an endianness?

Now when you understand the general concept, it is very important to find out how to implement this in reality. Let's think about how to transfer information between processes. How to send or receive data within a node on a computer network?

Reflect further on these questions and you will discover how to develop software for computer networks.

- What is a socket?
- What is the concept of socket programming?
- What is the difference between IPv4 and IPv6?
- What is the difference between a process and a thread?
- What is an Inter Process Communication?
- What is a port in computer networking?
- What socket functions exist?

Security and privacy are also crucial, given that no one would want their private conversations exposed. One of the main secrets of the Telegram app's popularity is its safety. So don't forget to give due attention to this topic.

- Why is there no password required for your Telegram account?
- How does authentication work in other messaging apps?
- How does an SSL certificate work?
- Which apps need to use cryptography today?
- Where must a firewall be located on a network?
- What is a hash function? How do developers use them?
- Which types of encryption are available today for developers?

And, of course, there is no team project without teamwork. So, answer these questions, which will help your team building process.

- What is the technical background of each team member?
- What technologies has every team member used before? What frameworks have you used?
- What are your expectations for this challenge? What is your deadline for this challenge?
- What tools will you use for teamwork? Jira, Asana, Trello, etc.?
- What each of you would like to do more?
- Who will be involved in the presentation of the product? Any ideas about the presentation?

GUIDING ACTIVITIES

The goal of this stage is to get the team to the point where it is ready to start designing a specific solution. Below you will find a list of consecutive activities to help you with this. Of course, you have to go through every stage of the challenge (including this one) with the whole team.

You already have experience working in a team. Now, it is time to reflect on the past and improve. What collaborative tools have you used before? Speak with your teammates about everyone's experience that can be helpful during this challenge. During this stage, remember, that the goal of every challenge is to develop a solution that means something in your community and is sustainable.
Let's start.

- You may want to get familiar with git submodules. But it is not necessary.
- Discuss the terms of the challenge:
  - messaging app
  - instant messaging
  - social messaging
  - client app
  - social networking platform

  - any other terms

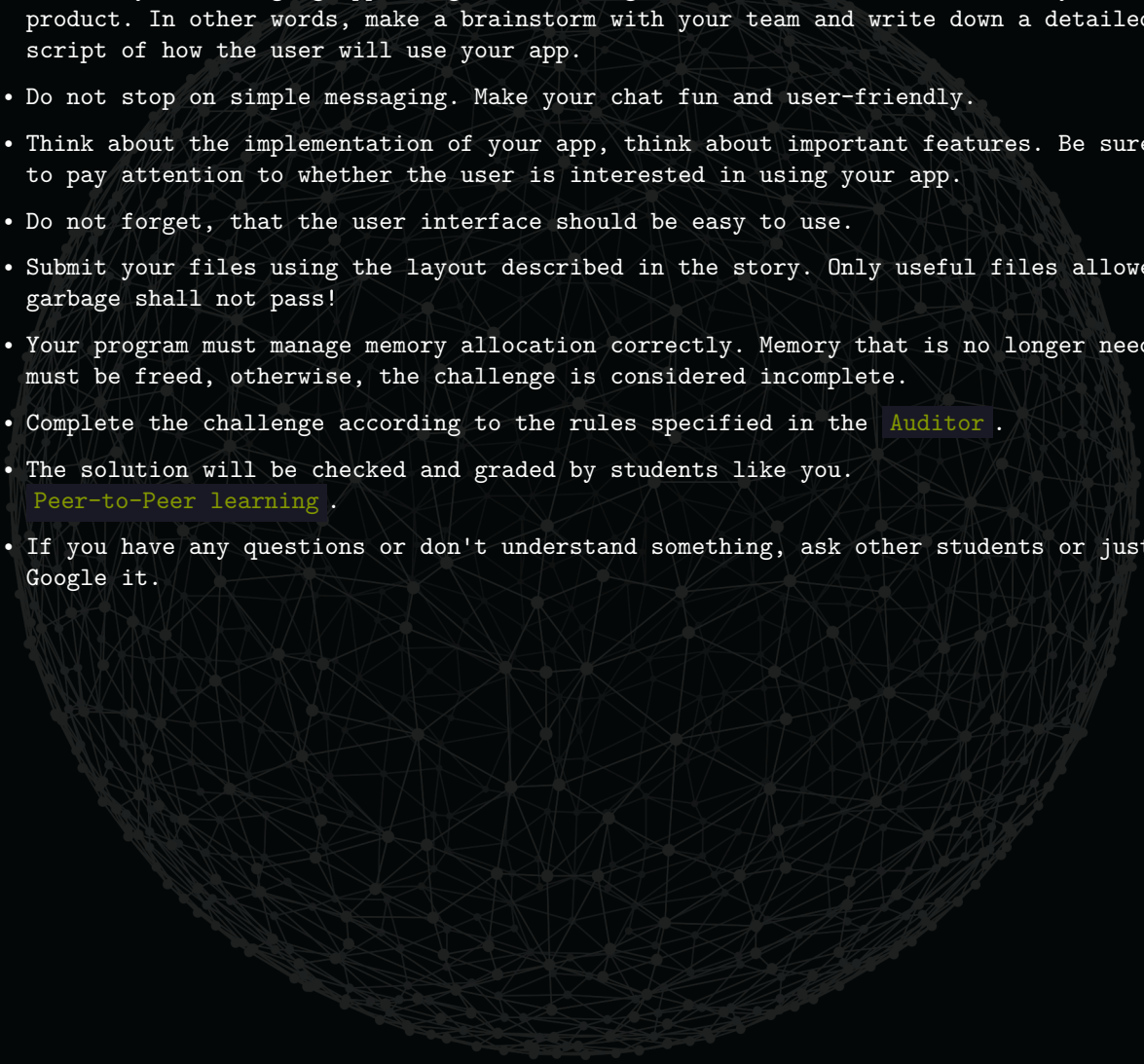
- Conduct a survey to find out what people like/dislike in messaging apps, what functions they use/avoid, etc.
- Think and discuss with your team which features are the most useful and necessary
- Brainstorm what makes texting a more preferable option to phone calls or personal meetings.
- Research existing communication apps. Which ones are the most popular? Why?
- Create a list of ideas and features for your product.
- Collect feedback on your shortlist of ideas, and select the best.
- Analyse information.
- Clone your git repository that is issued on the challenge page.
- Install `git-flow-avh`, create a `develop` branch and start pushing new branches with features.
- Only files that were pushed into the `master` branch will be evaluated.
- Begin from simple data transferring, without GUI for a client, just a basic console program to send and receive any kind of data. For now, your main task is to find out how `sockets` works.
- If your team wants to create a simple GUI then take the time to create design examples for your future client app.
- Implement some `tests`, it saves you time in the future.
- Develop the solution.
- Based on the information received, contribute to your product. Make improvements to your product continuously. Add some features such as calls, stickers, group chat, etc.
- Test your code.
- Discuss with your team how can you improve your product based on collected feedback. Make conclusions.

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.

- Challenge has to be carried out by an entire team.
- Discuss with your team the plan for implementing the chosen idea.
- Despite your ambitious plans, keep in mind that your project must be compiled in less than 30 seconds. Otherwise, the challenge is considered failed.
- Research the tools and technologies that will help you realize your solution.
- The challenge must be performed in `C`.
- This time, each line of text in your code must be at most 100 characters long.

- In this challenge, you can use `git submodules` for adding your `libmx` and open-source third party developed libraries like `json-c` in your solution. But make sure that
  - chosen libraries are public and anyone can clone them
  - your project can be built with one `make` command
- Discuss your messaging app design. Create logic for how the user should use your product. In other words, make a brainstorm with your team and write down a detailed script of how the user will use your app.
- Do not stop on simple messaging. Make your chat fun and user-friendly.
- Think about the implementation of your app, think about important features. Be sure to pay attention to whether the user is interested in using your app.
- Do not forget, that the user interface should be easy to use.
- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Your program must manage memory allocation correctly. Memory that is no longer needed must be freed, otherwise, the challenge is considered incomplete.
- Complete the challenge according to the rules specified in the `Auditor`.
- 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

BINARY

uchat, uchat_server

GENERAL

We hope you already understand the idea of the client-server architecture.
uchat is composed of two parts. The first part is the server. The server must handle requests from all client apps and must do it really fast. The second part, as it is easy to guess - the client. The client is nothing more than a beautiful wrapper for the simple user, it should never make real work, only request to the big brother (Server app).

Your goal? Create two programs in C:

- The first is named uchat_server which takes an integer argument, network port. If no arguments are given server stops and displays usage. It must be running as the daemon. At startup, it'll display process id of this app
- The second one is called uchat which takes two arguments, server IP address and port. It must establish the connection with the server app

The server part

- It's more appropriate to implement a special type of server, because parallel operations are the only way to complete this challenge.
  Read about concurrent servers. It could be useful.
- uchat_server must work as a daemon, listening to the specified port. Each connection must be correctly established. Maximum limit of users is set only by the operational system.
- Chat history must be always available for the clients. Even after client or server reboot.
- All necessary data must be saved into the database, you can use simple realization like SQLite or any other you like. But make sure that database deployment is performed with "a single click" and does not require any actions to configure the environment during the assessment.

The client part

- There must be basic authentication by username and password. But we encourage you to get a better understanding in this topic and implement cool encryption algorithms.
- If your uchat app lost connection with the server, you must manage reconnection. Appropriate messages must be displayed, also history mustn't be loaded or displayed again.
- The program must be able to edit and delete messages after they have been sent.
- The recipient must see the senders' names, the messages and the time when they were sent.

Architecture

Architecture gives a basis for analysis of the software system's behaviour before the system has been built. It helps to reduce risks and chances of failure even at the beginning

of development. We encourage you to use 3-tier architecture in this challenge because it helps to organize data storing, system of connectivity and security services easily. Also, it is clear that functionality must be logically divided into levels, because all these architectures, in theory, are useless without proper practical implementation.

Security

There is always room for safety improvements. Thousands of credit cards are sold and bought on the darknet everyday. Modern cryptography has come a long way, but scammers have gone even further. Have you heard that 267 million Facebook users IDs and phone numbers were exposed? So we kindly ask you to do your best to protect our data better than everyone else.

User interface/User experience

To develop a great user interface (UI) it's not enough to use advanced C graphic libraries. You should know that it just gives you a variety of abilities to implement better GUI part. But if you want your product to be successful, you should improve the user experience (UX) part. It means how easy it is to use your app. You could implement usability tests to investigate what is the best UI for your product. It helps to see what faults or missteps could real users have when they use your app. Also, these tests could be implemented during the project lifecycle, even before it is launched.

Features' importance

It isn't easy to make a user download your app. You need to interest people in your product! It's time to add some exciting features to your messaging app.

We are absolutely sure that you have used group chats to coordinate work and, of course, send memes. In addition to text messages, it would be cool to send files with different formats, for example, music, photos and stickers. Also, people like using emoticons. And, think about contact or messages search.

By all means, we want to tell you that in this challenge you can implement whatever you want. Emphasis can be placed on anything. If you are paranoid, create something safer than Wickr or Signal. If you like multitools, develop something with wider functionality than WeChat. Everything is in your hands.

Good luck!

Evaluation

What you need for assessment

- Solution on your git repository
- Presentation of your product

Presentation is an important stage in the process of product creation.
You must be able to defend your solution in front of an audience.
First impressions are critical.
Proper preparation is vital to presenting your product in the best light possible.

The objective of product presentation can be different depending upon the target audience and the presentation should be adjusted accordingly. It is important to know your audience and why they are interested enough to hear your presentation.

You will need to create a presentation that you will use during the defense, for other students. The presentation can be in any form convenient to you.

Examples of presentation tools

- Google Slides
- Keynote
- Prezi
- YouTube or any other video-sharing service

In your presentation you should tell about the experience of working on the product. From the moment you put together the team, up to, in fact, the stage of creating the solution and preparing for the presentation. You must present your product as if you want to sell it to investors who are interested in making a huge investment in the development of your product.

The product is one of the most important keywords here. Place a small piece demonstrating what you've done in your presentation. This way, the audience will see not only your brilliant presentation skills, but also some proof of your hard work.

The optimal time of the entire presentation is 7-10 minutes. It is pretty enough to make the audience interested in your product, show all the necessary features, and not to overload them.

Discuss with the team how you will present your solution. It is very important to prepare for a good presentation. So decide how you will divide the presentation content among all of your team members.

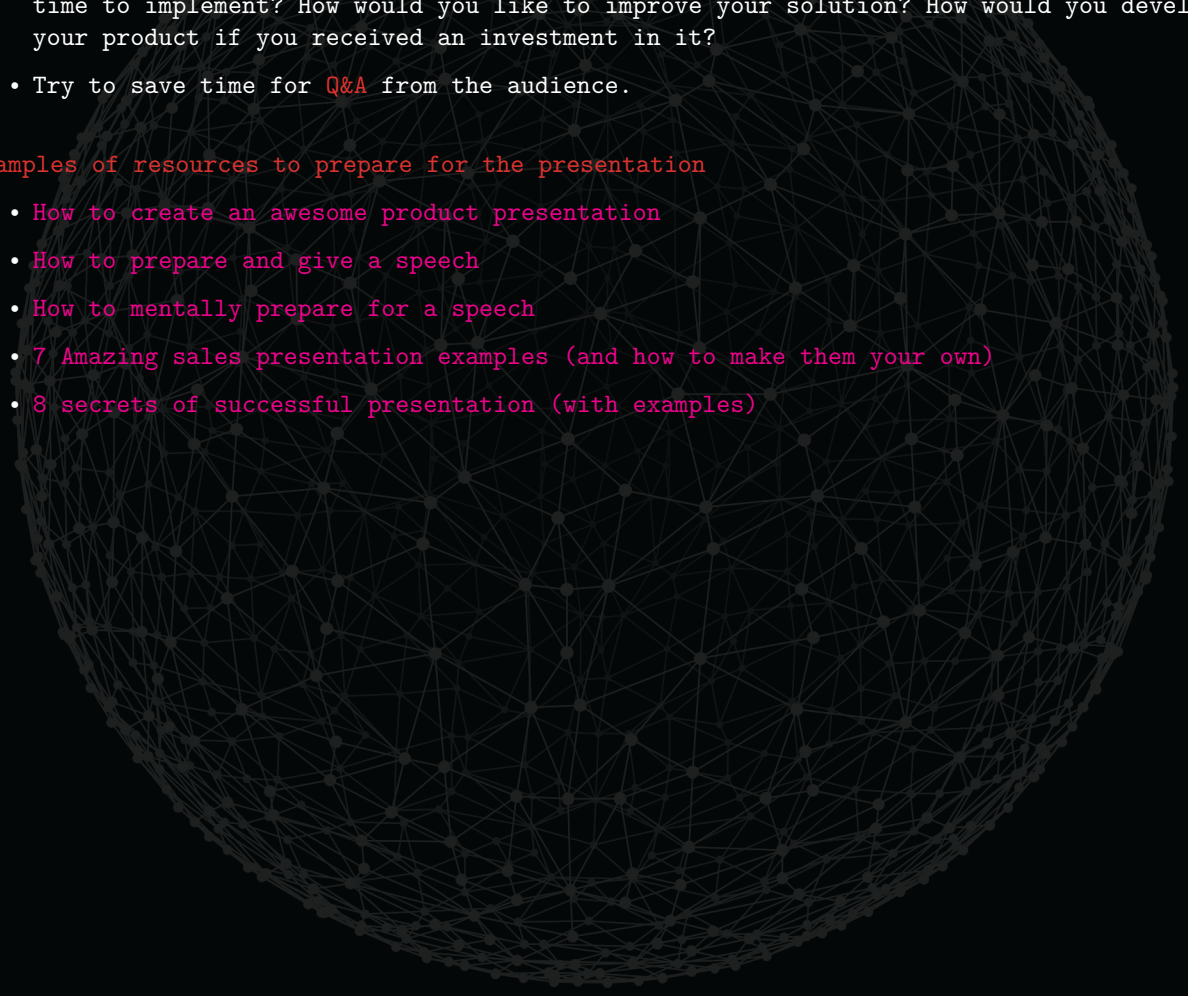
A bunch of advices

- Make a quick introduction of your product. Tell about your team, idea of your solution, why you chose it.
- Tell how you identified the solution. What kind of feedback have you obtained about the selected product topic, what were the expectations of the users, how did you take into account user feedback?

- It will be interesting if you tell how you worked on the solution. What influenced you while working on the creation of the product. What purpose did you follow?
- Tell what techniques and algorithms did you use? How did you use it? Why such a choice? What difficulties did you encounter during its development?
- Don't forget to show your product in action. Record the screen of how you use your program. Make it look like a trailer of your app. To pay attention to the most essential moments of your solution, capture the screen and use it in your presentation. Remember that you are limited in time.
- Describe how you see the further development of the product. What did you not have time to implement? How would you like to improve your solution? How would you develop your product if you received an investment in it?
- Try to save time for Q&A from the audience.

Examples of resources to prepare for the presentation

- How to create an awesome product presentation
- How to prepare and give a speech
- How to mentally prepare for a speech
- 7 Amazing sales presentation examples (and how to make them your own)
- 8 secrets of successful presentation (with examples)



Reflection

The next stage of your work is reflection. During the reflection, you are doing a retrospective of all work on the product. This allows you to understand the experience gained, consolidate your knowledge, and understand how to overcome the difficulties in the future.
Do not delay, gather the next day after the presentation, and discuss your experience and what you have learned while working on the product. Make a brainstorm and reflection on the results of this challenge. Take note of the topics discussed.

Example of topics

- Discuss all stages of your product. From team building to product presentation.
- Discuss your teamwork, interaction in the team. What were the issues of your teamwork? How can they be solved? What did you like the most in your team organization process? Did you use any approach to team management and software development?
- Discuss the competencies of each team member. Discuss positive and negative points. What expertise did you improve? Be open and truthful. This will help all of you to become better in the future.
- Discuss the technical implementation of your solution. What mistakes did you make while developing your product? How can you avoid them in the future? What new knowledge have you gained while working with selected technologies? What will you use again upon development?
- Discuss testing and user feedback. What kind of improvements would you make? What mistakes were made in the process? What is worth paying more attention to? What was the most attractive fo users? How could you do it much better?
- Discuss the preparation for the presentation. What did you not pay attention to? What will you take into account in the future? What part of the preparation took a lot of time?
- Discuss the presentation of your product. What you have not considered? What were your advantages? What could help you get your presentation better?
- Discuss how you can share your experience with the world. It will consolidate your knowledge and experience, so take it seriously.

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Some activities that can help you in the publishing process:

1. Prepare yourself a cup of something warm. We recommend some hibiscus tea, known as carcadè.
2. Get into a comfortable armchair. Turn on your favorite music. We were listening to twenty one pilots while working on this challenge.
3. Just start writing and don't stop. At first you will resist it, but then you'll like it. Just stay calm. We all went through it.
4. Use your presentation as the backbone of the publication you're writing. It can help.
5. Describe the key ideas of your product. What prompted you to focus on them for your solution?
6. Share your team work experience. How did you communicate with the team? Did you have any meetings?
7. Describe the technology stack you used. Why?
8. Analyse what you have actually accomplished. Was there something you planned to do, but didn't manage, or ran out of time?
9. Explain how you would improve your product? What is your hypothesis about the professional development of your product?
10. Now relax. Lean back and take a deep breath.
11. Analyze what you went through, what experience you have gained. What conclusions did you make from this challenge?
12. Are you ready to continue?

Helpful tools:

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

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.