Project Assignment

Messaging application

- Click through the prototype
  - Go to **https://www.fluidui.com/editor/live/preview/cF82N211bDdjMER4Y0F3UERBWFdsT1RMOWxxdVAwSFUzTQ==**
  - When prompt appears fill **any name** and **email**
  - If you do not see Sign up page, click ***Restart preview*** (on the left middle part of the page)
  - Prototype is mean to show intentions and features (not design) and is not complete

- You can also find some inspiration in messaging tools like *Slack* or *MS Teams* or *Skype* or *HipChat* or *Bitrix24* or *Roctek.Chat* or *Jostle* or *Moxtra* or *Azendoo* and others

Feature set

Channel management
- Create new
- Delete exiting
- Change name
- Invite user(s)
- Ordering

Attachments / images
- Options:
    - Inline in rich text
    - Attach to message
    - Attach to channel
- Images should have preview thumbnails

Messaging in channels
- Send message
- Delete message
- Profile picture next to message
- Message up/down voting
- Rich text editing experience
-         Font size, font color, triple emphasis, links, …
- Annotate existing user

Profile management
- Upload avatar
- Change displayed name
    - Use email strictly for sign in and invitations

Other requirement

- Keep your project's structure similar to the reference implementation
- Write tests for
  - Reducers
  - Thunk actions
  - Utility functions
- Write code based on SOLID, YAGNI, DRY, KISS principles
- Follow best practices (these can usually be found in the documentation of respective packages)

- Provide your users with indication on asynchronous operation progress
  - a loading spinner of some sort, …
- Try to find a way to deliver new messages to your users even when they do not interact with your application
  - You are provided with REST API, so new messages are delivered from the server only on your application's explicit request (→ no WebSocket involved) → messaging is not expected to be instant

Where to start

- Create UI using react & redux
  - Use whatever IDE you want, but we recommend WebStorm (free for students)

- Connect your UI to the server
  - See https://pv247messaging.azurewebsites.net/help/
  - Create an appId using POST on /api/app
  - To create a login/register login:
    - PUT request on /api/{appId}/{new-email}
    - Add Authentization attribute to the request header (use bearer authentication: { 'authentication': '{bearer} {token}' })
    - place any data your application needs to store into customData field
  - Feel free to browse REST API source code as well: https://github.com/KenticoAcademy/PV247-API

- Do not store/use/send any personal or sensitive information over the API

Problems or Questions or API bugs

- Browse through reference code or API source code
  - https://github.com/KenticoAcademy/PV247-API

- Use Discussion groups – courses application in IS
  - Read through the existing threads first, please
  - Attach a link to the problematic code in your repository (if applicable)

Submissions

Milestones

- 21. 10. 2018 Interactive wire-frame
- 11. 11. 2018 Implemented features to work locally with redux store
- 2. 12. 2018 Client integrates with backend, async actions are implemented
- 16. 12. 2018 Tests, implemented Rich text capabilities

Deadline
16. 12. 2018 23:59

- Insert a link to your public GitHub repository into *Project* homework vault ("odevzdávárna")
  - Link your UČO/names with used GitHub accounts. Only two GitHub accounts are supposed to commit to the repository.