The problem
Requirements

- Stable pairs
- Friends
- Distribution of skills
**Architecture**

- **User**: GET, POST, PUT, DELETE
- **Group**: GET, POST, PUT, DELETE
- **Backend API**:
  - `/sendmail`: POST
  - `/authenticate`: POST
  - `/assign`: POST
- **Frontend**:
  - Post results of the algorithm to the API
- **Algorithm**:
  - `/run`: POST
  - Post the configuration of students and groups to the algorithm
Timeline

Today

Algorithm

Backend

Setup

Algorithm

Frontend

Deployment
just mention it, pick some of the 12 “factors” (the ones we think are interesting) and explain what they are and how we implement them

marc sounds like a good idea

gorgy not sure we do all of that but it’s nice yeah

bram well, we can try to do as many as possible like, point 5 does not really apply to Node

georgy 7 does that apply to node? I’m not really sure.

bram it does, but if we go for a docker environment, then you have to do it that way

gorgy Yes but in a docker environment you can also have tomcat right?

bram you could

the cleanest (imho) way, is defining a port you’re listening to in the container inside your app. Then you have Docker forward that port to expose it to the outside doing it that way means your app must be listening to a port and Docker is your middle layer

georgy I’m at CLC
Travis CI
Questions?
Milestones

- Algorithm research
- Basic project structure (API)
- Authentication (API)
- Working algorithm
- Front-end user
- Email integration (API)
- Front-end supervisor
- ...