Software Project
LTAT.05.005
Time Management and Teamwork

Coordinator: Kuldar Taveter
Mentors: Yar Muhammad, Alexandru Gutan
Not an ordinary course

• Very few “lectures”
  • Introductory lecture and lecture about requirements + two demo sessions
• Lots of teamwork (about 150 hours per member, 600 per team)
• Lots of mentoring – weekly meetings with mentors starting next week
  • Mentors will clarify what you need to achieve and by when, they will help you resolve conflicts and mediate problematic situations, and will assess your deliverables
• Lots of online communication…
  • Please make sure you are registered in the Slack workspace https://tvp2020.slack.com
  • Preferably, use Slack channels to communicate within your team
(Almost) everything you want to know or ask should be here...

- Course web page: https://courses.cs.ut.ee/2020/tvp/
- Slack workspace: https://tvp2020.slack.com
- TinyURL for joining the Slack workspace: https://tinyurl.com/y67xmfr3
Development and collaboration infrastructure

• Every team must set up a development/collaboration infrastructure

• Min requirements:
  • Version Control System (VCS), e.g. git
  • Issue tracker
  • Wiki
  • Continuous Integration

• GitLab is strongly recommended as the all-in-one solution for CI/CD and project management (Github or BitBucket, but extra complexity is on you)
  • It’s your choice, you set it up and then grant access to your mentor

• For CI of mobile apps you can use GitLab or NeverCode.io (free tier)
Dev and collab infrastructure (cont.)

• We can provide:
  • A Linux Ubuntu or CentOS bare-bones virtual server with 2 cores, 2 GB RAM, 20GB Disk in University of Tartu’s cloud (available with VPN access only)
  • If you need more capacity (CPUs, RAM), ask, but not sure we can provide it
  • If you need external IP address (to show to your customer), please ask upfront
  • Drop me an email to make your virtual server request
Dev and collab infrastructure (cont.)

You can also use another deployment infrastructure, e.g.

- **DigitalOcean** (recommended) – simple, reliable for Docker containers ($100 credits with the GitHub education pack)
- Any of the major cloud providers: AWS, Google Cloud, Azure (check “always free” or trial services)
- Heroku (Supports most programming languages as well as Docker), slow machine boot and downtime on free tier
- Engine Yard for PHP/Ruby/Node.js
Dev and collab infrastructure (cont.)

• Docker for CI/CD is strongly recommended
  • Normally, your CI/CD process should end in a deployed/executable application
  • However, producing a docker image in Docker Hub and “manually” pulling it into your deployment server on UT’s cloud is OK
Working Principles

1. Transparency
   • Anything that is not in your dev and collaboration infrastructure does not exist
   • Any server-side application that is not running on a server does not exist.
2. Traceability

2.1 Create an issue
2.2 Create a new branch
2.3 Create a new merge/pull request for that branch and link it to the issue
2.4 Every commit to that branch would go into the merge/pull request and will be automatically linked to the issue

Issues will be linked/grouped in milestones which will be modelled after the requirements (documented in the Wiki).
3. Reproducibility

- A person having ordinary skill in the art (PHOSITA) should be able to build and test your app from the source code with reasonable effort. Your mentor is a PHOSITA.

- Docker is strongly recommended (where applicable) to ensure the portability of your solution and avoid the “it works on my PC” excuse.

4. Continuous Integration and Deployment (CI/CD)

- Every commit/push on any branch should lead to a new tested version of the system

- Every commit on master branch should additionally lead to a deployed version of the system

Tip1: Use Docker-compose locally to run unit tests quickly
Tip2: Use GitLab-runner locally for performance intensive tasks or to save CI/CD minutes by running the pipeline locally.
What to do now/next?

• Choose your project & contact your customer
  - Choose your project and form the team by Friday 4 September 8 AM
  - The contact details of the customer should be available in the proposal, as a rule. If not, please let me know.

• Agree on a first meeting with customer

• Prepare for your meeting – read the project description, prepare your meeting, do some searches online to get info, check for possible approaches, think about possible requirements, prepare questions

• Read the grading criteria (at least for iteration 1)

• Set up your dev and collab infra

• If possible: discuss how to use the collab infra – who monitors/oversees issue tracker, who writes to Wiki what and when, what is your Git workflow (e.g. branching and merging policy)

• Reminder: Iteration 1 deadline is 21 September
Iteration deadlines

• 4 September: teams formed
• 21 September: 1st iteration (two weeks)
• 12 October: 2nd iteration (three weeks)
• 9 November: 3rd iteration (four weeks)
• 7 December: 4th iteration (four weeks)
Other deadlines

• 21 October: Intermediate demo day
• 6 November: Will send you a message with information about peer-review
• 13 November: Deadline for selecting a project to peer-review
• 23 November: Peer reviews due
• 7 December: Peer-review responses due
• 16 December: Final Demo Day
Remarks

- The consultation sessions will be held as two parallel Zoom sessions with 20-minute slots for each team on Wednesdays at 12.00-16.00.
Past examples of very good projects

- https://github.com/leelokas/vacation_tracker/wiki
- https://github.com/leelokas/vacation_tracker
- https://github.com/vladislavivanistsev/SuperCap/
- https://github.com/alajal/license-management/
- https://github.com/L6mps/LasaLaraAP/