Software Engineering 2022-2023, Prof. Nuo Li

MID-TERM FEEDBACK MC-MESSENGER

1. PROJECT INPUTS

Documentation : https://github.com/Scherrik/se_mcm/tree/main/docs

Source code : https://github.com/Scherrik/se_mcm

Scrum board: https://se-mcm.atlassian.net/jira/software/projects/SM/boards/1

Blog: https://semcmessenger.wordpress.com/

2. GENERAL FEEDBACK

→ A lot of discussion and well documented.

→ Stored all documents and blogs in GitHub – very good!

→ Very good development progress

→ Suggestion:

- How about security? Any thought to improve it?
- Think about how to implement CI/CD. Your project has a good basement to practice
- Based on the current standard, your team may get 13 (presentation) + 78 → 91 Points[1] (out of 96)

3. MID-TERM CHECKLIST

- Project handout (the same handout for your presentation)

→ Done.

→ Architecture style/decisions and major arguments for this choice is missing.

- RUP SRS document (Software Requirement Specification)

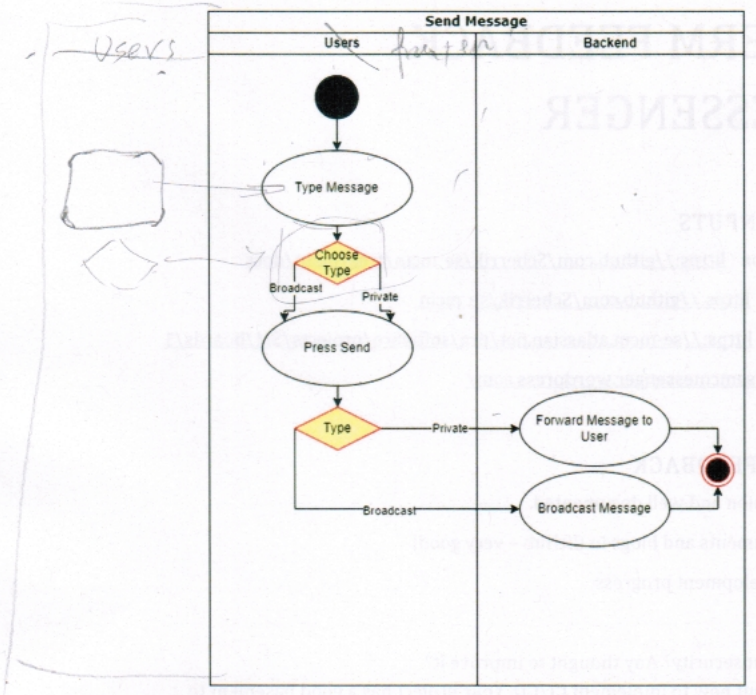
→ Well done!

→ UML activity diagram has no eclipse shape ... activities are rectangular with round corners ....

---

[1] Based on the assumption that your team will do a good job for the tasks in Semester 4

→ If no matter choose which type, the next step is press send, why do you distinguish them here?



- Include your project vision into SRS
- Tipps: Extract information from your utility tree to fill in the non-functional requirements.

- RUP SAD document (Software Architecture Document)
  → The readability of the description in ASR is low, which need to be improved.
  - Tipps: Embed the "Use-Case-Realization Specification (UCRS)" into Section 4, if they fit – check the explanation of Section 4.1.
  - Tipps: Embed your class diagrams into Section 5.
  - Reminder: make the whole document readable as one piece.
- Links to your GitHub and Scrum board
  → ~~I cannot find your code repository~~...
- If you cannot authorize me the access to your GitHub and/or Scrum board, please submit a separate report including the following information:
  - *Full list of your user stories*
  - *Description of your sprint setup*
  - *Burndown charts of each sprint*
  - *Closed tasks for each sprint*
  - *Explanation of how your team organize project artifacts*

- *Screenshots of your Git branches*

4. FINAL CHECKLIST

Understand business need - 25% → well done. Some small errors, see comment above.

- Vision
- Scope
- Use cases / user stories – at least 5 high-level use cases reflecting the main functions of your software (excluding user login)
- Software requirement specifications and non-functionals
- Relevant UML diagrams

Project management – 25% → Well done! There are some open tasks in the last spring, which should be done already.

- RUP (Rational Unified Process) workflow
- Gantt chart, GENERATED not done by hand (given all the deficiencies that we know)
- Long-term planning (including hours/team member breakdown)
- Scrumming and iterative process, burndown charts
- Cost estimation (Function points or Storypoints)
  - Comparison of estimation vs. real time spent
- Artifacts management/version control, including source code and documents
- Risk management (live) document. → comes later

Technical ability - 25%

- Demo -- Must have more than Login/Registration and match your Scope (live demo is in the presentation, or submit a demo video. In any case, the documentation should include demo screenshots)
  → Very good progress!
  → Plan for your final demo and focus on the key features.
  → Any thoughts about enhancing security?
- Design documents, including UML diagrams
  → well done.
- Architecture overview (UML drawings and summary of your highlight, why do you choose this architecture)
  → Well done, but please improve the description in ASR
- Summary of applied techniques, platforms, tools, etc.
  → Make a detailed summary including why you chose them in the final report
- Continuous Integration / Life cycle management (automatic deploy etc.) setup description → comes later

        - Degree of automation for example, the last push of the day can trigger all kinds of things, metrics, tests, deployment, IDE, git and PM should be integrated.
    - Source code (link to your repository – must be accessible by me)

Quality - 25% → most of them comes in Semester 4

- Design patterns
- Clean code principles
- Testing (test cases, test reports)
    - Test Cases
    - Test Plan
    - Test log (for example screen shot in red/green)
    - Test coverage report
- Metrics
    - Explain 2 metrics, and show how code change improves them.
    - Show code samples when explaining metric.
    - Explain why some parts of code have bad metric but no need to change
- Security concerns