Building Social Network

The project aims to design and implement a software product using state-of-the-art tools and technologies in the software industry. The required product should mimic some functionalities of **Reddit** [Web API]. You are allowed to implement another project with the following conditions [after a discussion with the TAs]:

1. Similar size.
2. Have an open API [This is required to allow both front-end and back-end developers to reduce dependency as much as possible]. If an experienced member can start working early to finish a design for the API as early as possible and all the team members accept this risk, this condition can be discussed.

Objectives

1. Software process tools:
    a. Task Management Tool.
    b. Version Control Tool.
    c. Documentation tools (API & Functional (code) documentation).
2. Concepts including:
    a. Back-end, front-end architecture.
    b. Design patterns: For example, MVC.
    c. Unit Testing.
    d. Restful API.
    e. Using a well-known programming model (Incremental [Sprints] recommended).
3. Implementing:
    a. Learning how to use package managers.
    b. Implementing and consuming Restful API.
    c. Unit Testing.
    d. Using recent technologies and appropriate stacks.
    e. Using unified code style for the back-end and another (can be the same) for the front-end.
4. Team-Work Skills:
    a. Designing and Managing Software Process.
    b. Dealing with large teams (compared to your previous teams).
    c. Separation of responsibilities & Smooth Integration.

Team members and Positions

1. Team Leader [1 member]

- Manage duties and tasks.
- Responsible for delivering the project deployed, integrated, and completely running.
- Deadlines Determination and announcements: deadlines should be early enough so that all integrations can be done and tested.
- Main communicator with TAs.
- Communication with other sub-team leaders to ensure everything is going as scheduled.
- Construct and manage a smooth and formal communication channel for the team.
-

2. DevOps Team [1 member + Team Leader]:

Team Leader + DevOps Engineer.

- Use the cloud or any other hosting service with HaaS (Hardware as services) [Check the TA for further info].
- Very quick deployments.
- Manage Linux-based systems. This includes multiple tasks such as logs, remote access, some automation, and scripting.
- Set up tools for monitoring and automation (e.g., Jenkins or Travis-CI).
- With the team leader: make choices of what to use, when to deploy, and other decisions related to the process. The DevOps team does not do that task in most companies, but this is for educational purposes.
- Auto-scaling on the cluster level and the services level [Optional]
- Use Kubernetes.[Optional]
- Dockerization (optional but recommended to ease the deployment process)

3. Back-end Team [4 members]

Sub-Team Leader + Back-end Software Engineer(s)

- Construct a complete requests collection with an API Client to send all requests with different values for significant parameters. [Example tool: Postman].
- Unit testing with coverage above 95%
- Auto-generate documentation for the REST APIs and the source code
- Authentication and Authorization for applicable API requests.
- Design and implement a valid database system [the concepts of "Migrations" and "seeds" should be used to quickly initiate and populate the database for easy and incremental deployment].
- Deliver an ER diagram and API documentation in addition to other tasks.
- Software Design (Requests Design).
- Integration.
- Any modern language is accepted; you can use Python, Node.js, Go, or whatever you wish. Using a framework is highly recommended.0
- Any modern database system is accepted, for example MongoDB.
- Expected work: Same size as the DB project. As a rough estimate, each member should implement one large module or two medium modules.
- Use Docker
-

4. Front-end Team [6 members]

Sub-Team Leader + Front-end Software Engineer(s)

- Unit testing with coverage above 95%
- Responsive [Will be tested on browsers with various screen sizes].
- Use AJAX requests whenever possible.

- Do a service to mimic the back-end responses (with fake data) to work without the back-end whenever needed. Alternating between the real and the mimic back-end should be as smooth as possible with global configuration modifications only. In modern frameworks, this concept is called Mock services and is usually implemented using the dependency injection concept.
- Software Design.
- Integration.
- Expected work per member: 5 mid-level pages with all relevant services. The page's size depends on the different components it has and the diversity of data it needs.
-

5. Testing Team [2 members]

Sub-Team Leader + Software Test Engineer(s)

- End-to-end testing (E2E):
    - For web: 90% coverage
    - For Android & Cross Platform: 90% coverage
- Stress testing

6. Android or iOS Team [3 members]

Team Leader + Android Software Engineer(s)

- Unit testing with coverage above 95%
- Integration.

7. Cross Platform Team [5 members]

Team Leader + Cross Platform Software Engineer(s)

- Unit testing with coverage above 95%
- Integration.
- The implemented product should work on Android and web and it should be deployed. You are also required to build it as a web app and a desktop app if the framework supports that.

- For all members:
    - Submit a weekly progress report to the team leader [Weekly date should be determined and submitted to the TA at phase 0].
    - Update the state of his tasks.
    - Push the code periodically (on time). No Major pushes are allowed.
    - Use the tools chosen at the first phase.
    - Mention at least 3 used Design Patterns.
    - Clearly understand the project's architecture and how work is integrated between different sub-teams.
    - Documentation:
        - All the used tools.
        - License allowance of your product; for example, if you use a tool that prohibits commercial use, you should indicate that your product cannot be commercial.
        - Code should be acknowledged whenever used from any external resource. Any copied code - without explicit acknowledgment inside the source code, the “Readme.md” file, and at the final documentation- will be considered cheating.

Project Phases

1. Phase 0 [Proposal] (week 3):
    a. Used Tools & language (All of them: front-end, back-end, data system [for example DB engine] and so on).
    b. Team members, Positions, and your company name.
    c. Preparing repo & Task Management Tool.
    d. Code Style for each team.
    e. Google drive link containing all deliverables; it should contain a directory for each deliverable, and deliverables should be uploaded each Saturday before 8 pm so that all the team can revise it before the deadline, which is 11:59 pm. If you were asked to change any deliverable, keep the original one and make a new version—for example, Phase 1 -> V1, Phase 1 -> V2, and so on.

2. Phase 1 [Proposal] (week 5) [5%]:
    a. Task division for **all tasks** of the project among **all team members** (Use task management tool if appropriate).
    b. System Design [With the help of Open APIs].
    c. [BE] ER Diagram, API documentation
    d. [FE] [Android] pages choices and assigning them among the teams should be delivered in this phase.
3. Phase 2 (week 7)[10%]:
    a. 20% of the project should be finished (each member should finish a part of each point assigned to his position as indicated in the positions part of this document).
    b. Progress report for each member and a combined progress report for the whole team.
4. Phase 3 (week 10)[25%]:
    a. The same as phase 1 but 50 % of the project should be finished, and a prototype should be deployed and running.
5. Phase 4 (week 13)[60%]:
    a. 100% of the project should be finished.
    b. All the other deliverables should be ready.
    c. Each sub-team should deliver 1 version for each diagram taught in lectures. For example, the use-case diagram.
    d. Deliver a presentation with a working prototype of your project.
    e. Deliver a version history sheet to illustrate different versions, releases, and all team members' contributions.

Deliverables

1. A document with the full names and IDs of all team members.
2. Progress reports (Individually and for the whole project) (Starting from Phase 2).
3. Meaningful Logs of Git repository, project management tools, and other used development tools (Starting from Phase 2).
4. Documentation. It depends on the team, as detailed above.
5. A fully integrated deployed working product.
6. Link to the source code Git repository. The TA must also have access to the repo.
7. [Final phase only] [Will be confirmed later] Demo video showing the product and all its functionalities. Time limit: 5 minutes.

8. [Final phase only] [Will be confirmed later] Individual progress videos to explain the contributions of each team member. Each member must record a separate video. Time limit: 2 minutes per member.
9. Demo or a Presentation, clarifying:
    a. The final product features and limitations.
    b. Future work.
10. A Document for:
    a. Final Work Division.
    b. Problems the team faced (technical and non-technical problems).

After the first phase, each group will be assigned to a TA and will submit the project documents to this TA through email. For each group, please put the email of a contact person to reach you easily. For support & delivery emails, use the following email subject template: "[SWE-F2022] ***". For example: "[SWE-F2022] Front-End Integration Problem".

Recommended tools & Suggestions

1. https://education.github.com/pack: includes Github premium, cloud, task tools, and others.
2. VSCode as an IDE.
3. Front-end and Back-end tools: must be recent. Front-End examples: Angular 7+, React, VueJS. Back-End examples: Python, Node.js, Go. Before choosing the tool to use, you should ensure that it satisfies the other requirements (such as Auto-Generated documentation, unit testing, and E2E testing).
4. Open-Source projects can help you to:
    a. Divide the project into modules.
    b. Monitor recent technologies.
    c. Learn appropriate coding styles.

General Rules

1. Because of hardware requirements, the iOS team is optional.

2. To work with a cross-platform framework, teams will change slightly depending on the platforms the team will deploy to.
3. There is a tolerance for two more or fewer members in the whole project team.
4. User Management is a must [Registration - Login - Forgot Password - Mail Verification - Change Password, etc.].
5. Add Explanatory files. For example, Release notes and Readme.md.
6. Pagination [Partial Retrieval] is a must whenever applicable.
7. Understanding the concepts, structure, and architecture of the used framework or language is very important in evaluation.
8. All teams should be synchronized; for example, if BE is working on Authentication, both FE & Testing should be working on the same part in the same phase with the "agreed-on" Requests' API. Communication is mainly the responsibility of the team leader and sub-team leaders.
9. All testing should cover direct and corner cases. Testing the effects of actions should be thorough. Naive assertions will not be accepted in unit testing or E2E testing.
10. [For testing] It is critical to make all selectors in a separate file to cope with FE GUI changes.
11. The connection between Back-end and either Front-end, Android, or iOS is the responsibility of the sub-team leaders; however, it can be handled by another member(s) if the team accepts. This task should be explicitly assigned in the first task assignment and should be submitted to the TA.
12. Cheating will result in zero in the project and negative grades of your other work grades.
13. Integration is graded for all related members.
14. Transparency is critical in evaluation. Lack of transparency may result in 0 per phase (without discussion). Each delivered report, including tasks, should be made with extreme caution and honesty. It's the team leader's responsibility to check each delivered report to make sure it reflects the **actual**, current state.
15. Related to the previous point: Sub-team leaders should check and approve any submitted code to make sure it works and comply with all practices, internal agreements, etc. The mistaken member will be penalized; however, the sub-team leader may be penalized more [up to 2x]
16. The last four days before phases 2, 3, 4 are dedicated to testing teams only. Pushes should be disabled. This is the responsibility of the team leader and sub-team leaders. Any break from this rule will be severely penalized. In exceptional cases, testing team members can risk permitting pushes for other teams (up to 2 days before delivery) with an official email to the team leader.
17. Huge commits are forbidden.
18. Pushes that do not work or do not comply with the practices chosen may be penalized.

19. The project is required to be delivered as a PRODUCT. In other words, it is not just about coding; it is about the complete software engineering process.
20. A Bonus may be given for top creative features or skills. There will be a bonus for the top students who help other teams learn, understand, and apply concepts they know through sessions, videos, or anything else related to the project.
21. Passing deadlines or unprofessional deliverables are penalized.
22. A FAQ document is released. Check both documents regularly before asking new questions. Both will contain details affecting your evaluation.