# P1: Let the games begin!

## 1. Introduction

In this project, you are to develop a software system to play a specific board game, henceforth referred to as the xGame. In addition to allowing users to play the xGame, there are some other features that the system should provide.

This is an incremental project to develop using agile practices. Ideally, you should follow the plan that you formulated in CW2, but things might change along the way. There will be three project milestones along the semester: two project updates (P1 and P2) and the final project delivery(P3). Each milestone includes a presentation and a set of artifacts that will count toward the milestone grade. While the rules of the xGame are immutable, the rest of requirements are always prone to change.

Professional conduct and academic integrity policies apply to all the activities in this course. Check the CS 414 website for more information.

## 2. The xGame system

The xGame system is a platform that allows users to play the xGame online against other players. The next transcript provides more information about it. If you have any doubt, ask for clarification.

> "What I need is something like a platform that allows users to play the xGame online. Anyone could register to this platform, for example by using an email, which would be unique for that user. To register, the person should provide a nickname (also unique, maybe public???) and a password.
>
> What can a user do in the platform? Mmmm. She could create a new match (so she can play it). Since she can't play by herself, she should be able to invite another user to join the match. Perhaps she could send more than one invitation, and then it would be something like "first come, first served", so the first user accepting the invitation will be the one joining the match??? Is that possible?? I guess a user also needs to be able to reject an invitation, so it would be nice if the user who sent it receives a notification.
>
> It would be cool if a user could be part of multiple games at the same time, though maybe she would want to quit from any game at any time? I think a user would also want to be able to unregister.
>
> The platform also needs to record the history of matches played by a user. Info like players, start and end dates and times, and end results would be useful, you know, to know who won or lost or if there was a tie. I guess info about abandoned games should be also recorded. All this info would be part of the user profile, which can only be viewed by registered users.
>
> The game play, well—the xGame has some rules that need to be followed during a match. Besides that, of course a game can't start until enough players have joined, and I'm guessing that after a match starts no other player should be able to join. Who starts the match? If I'm not wrong, that should be specified in the rules of the game. Otherwise, the user who created the match would be the one making the first move. Mmmm, the system should be able to determine whose turn is it... according to the rules, right? Meaning, a player can only make moves when it's her turn...

allowed moves, that is... the rules. What else? Oh right. The state of the matches should be saved in some way, so the user can play whenever she wants. My guess is that users won't be playing the whole time, so for example, a user would make a move whenever is her turn and log out, and after a while she would come back and check if the other player made a move and it's her turn again. Asynchronous matches, I think that describes it. The system needs to know when a game is over and should let know the players who won or lost. All according to the rules."

## 3. Development methodology and practices

As mentioned in CW2, although it is difficult to follow a strict agile methodology to develop this project, the idea is to apply as many agile practices as possible. When viable, you will follow the plan that you formulated in CW2. However, if conditions change along the term, you are encouraged to change the plan as needed. Remember that agile development values responding to change over following the plan.

## 4. The Manifesto for Agile Software Development [Beck'01] states:

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, there is value in the items on the right, we value the items of the left more.

You are to work as a Scrum team. Since the term project is an academic exercise, the TA and the instructor will act as product owners (POs), and the scrum master will be selected by each team. The team will carry out all the scrum ceremonies for each sprint. If you need the PO to be present during a ceremony, the scrum master must make the necessary arrangements.

The project presentations scheduled for this term are project updates (an academic exercise) rather than scrum ceremonies.

You are required to use a Kanban board to keep track of your progress in terms of tasks. Unit testing is another mandatory practice. Other kinds of testing, continuous integration, and continuous code quality assessment are optional but strongly encouraged.

## 5. Requirements engineering, software design and implementation

Since the term project is an academic exercise, you are to flesh out the requirements listed above as user stories. Moreover, you are to break user stories into tasks.

A user story is a short description of functionality told from the user's perspective. This description is complemented with acceptance criteria that can be used to determine when the story implementation is complete. User stories and acceptance criteria are defined by the PO. User stories are assigned with a development estimate (by the development team) and a priority (by the PO).

You are to use CRC cards and class diagrams to sketch the design of your software system. As soon as you have defined the sprint backlog and a design sketch, you can start addressing the sprint tasks. You are required to keep a traceability matrix between user story tasks and classes.

A traceability link matrix (TLM) is a matrix storing links between artifacts. Let's assume that we are interested in knowing which classes in a software system implement user story tasks defined for that system, i.e., the links between tasks and classes. Then, each row in our TLM will represent a task, and each column will represent a class. A mark in the cell intersecting the task US-1A and the class "MyClassB" indicates that the class "MyClassB" is directly involved in the implementation of the task US-1A, as shown next:

| | MyClassA | MyClassB | ... | MyClassZ |
|---|---|---|---|---|
| US-1A | X | X | | |
| US-2B | | X | | X |
| ... | | | | |
| US-nY | X | | | X |

## 6. Development environment setup

There are only a few restrictions in the tools and technologies used to develop the project.

- The project must be stored in a GitHub repository. Name the repository after your team's name as follows: cs414-f21-teamname. The repository should be used to store and track changes to the source code and any other document generated during the development of the project (i.e., development artifacts, presentation slides, etc.).
- The system logic must be written in Java. The type of UI and data storage are up to you, i.e., you can choose any technology you want to implement these components.
- Extra points will be given for using continuous integration, continuous code quality assessment, and testing techniques other than unit testing.

## 7. Project deliverables

There are nine deliverables for the term project:

1. **User stories and tasks**. They should be uploaded to the GitHub repository. Find a good way of doing so, e.g., in a PDF or MD file.
2. **Kanban board.** A weekly screenshot of the Kanban board should be provided. Find a good way of doing so, e.g., in a PDF or MD file.
3. **Design artifacts.** This document includes the CRC cards and class diagrams. Find a good way of doing so, e.g., in a PDF or MD file.
4. **Source code.** The code should include the JUnit test cases. Well-documented code will be rewarded.
5. **Development manual.** This document should describe how to set up the development environment to work on the project, how to run the system as a developer, and how to run the tests (put yourself in the place of a newcomer—what are the necessary steps for her to start working on the project?).
6. **Traceability link matrix.** This is a spreadsheet document showing the traceability links between user story tasks and the code.

7. **Output of scrum ceremonies.** Make sure you take notes during the sprint review and the sprint retrospective. Find a good way of sharing your notes, e.g., in a PDF or MD file.
8. **Presentation.** The progress on the project is to be presented during the due date lecture. In addition, you are to present a brief description of the **xGame**, as well as any process/product decision you have made. You are also to present the results of your scrum ceremonies. The slides of the presentation should be stored in the repository.
9. **Peer evaluation** (individual). Each team member is required to fill out the peer evaluation and submit it through Canvas.

### 7.1 Project milestones

For each project milestone, create a wiki page in your GitHub repository. Name the wiki pages after the milestone identifier (i.e., P1, P2, P3). Link the deliverables 1 to 8 in the respective wiki page when the milestone due date comes. Consider that not all the deliverables are required in all milestones. If a deliverable has not been started yet, mark it as "TBS". Check the rubric of each milestone for more information.

If you use development practices other than the ones required, make sure to mention them and link the appropriate resources.

The grades for each milestone will apply to the deliverables tagged with the milestone identifier in your repository. The tag should be created before the lecture (i.e. 9:00am) on the due date.

## 8. Notes

- Delivery dates associated with each milestone will be verified in the repository. Late work policies apply. Once the repository has been created, add the instructor3 and TA4 as collaborators.
- Instead of making assumptions about the requirements, talk with the product owner.
- Points will be deducted if:
  - The instructor and TA are not added to the GitHub repository.
  - You are late with the submission.
  - The submission requirements are not met.
- You will not receive credit for this assignment if:
- You do not submit the deliverables.
  - You do not present during class.

## 9. P1 rubric

| Item | Points |
|---|---|
| User stories and tasks | /2.5 |
| Kanban board | /1.0 |
| Design artifacts | /2.0 |
| Source code | /0.0 |
| Development manual | /0.0 |
| Traceability link matrix | /0.0 |
| Output of scrum ceremonies | /1.0 |
| Presentation | /2.5 |
| Peer evaluation | /1.0 |

| | |
|---|---|
| *Total* | /10.0 |

## 10. Board game resources

### 10.1 Extinction chess

- Extinction chess, Wikipedia
- Description & Rules for Extinction chess, The Chess Variants Pages
- Rules explained, Chess.com

### 10.2 Legan chess

- Andernach chess, Wikipedia
- Legan Chess Rules and Exception, BrainKing
- Legan chess Updated Rules, YouTube

### 10.3 Plunder chess

- Plunder Chess, Wikipedia
- PlunderChess®. 2020., Official Website
- Plunder Chess, The Chess Variants Pages

### 10.4 Portal chess

- Portal Chess, Wikipedia
- Porthole Chess Rules, Mark Hidden
- Portal Chess: David Howe version, The Chess Variants Pages
- How to play Portal Chess!, YouTube
- Portal Chess Updated Rules, YouTube

### 10.5 Omega chess

- Omega Chess, Wikipedia
- Omega Chess Rules, The Chess Variants Pages
- Omega Chess (open the link and switch language to English)
- Omega Chess 960 – Chess Variants Ep, YouTube

## 11. References

[Beck'01] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J., 2001. Manifesto for agile software development.