U (U) Not possible to run. Project to small, to few use cases, design to strange/bad (hard to understand). Documentation missing or poor. Not possible to trace any process from agendas. No or bad presentation.

Grade 3 (G) A project fulfilling the base requirements for all grades. Which means:

- **Functionality;**
  - The app is working
- **8-10 use cases implemented and working.**
  - 13 use cases are implemented and working
- **Simple but functional GUI.**
  - The application has an advanced GUI
- **Inhouse MVC.**
  - The application uses an inhouse MVC design
- **Clean implementation of at least one subsystems.**
  - Examples
    - AssetHandler
    - LevelBuilder
    - CollisionHandler
- **Application uses interfaces to reduce dependencies.**
  - As much as reasonably possible, we have coded against interfaces and not concrete implmentations
- **There are some usable tests.**
  - The application includes nine tests, which covers 100% of the model
- **Documentation is short but correct (and in sync. with the application).**
  - Documentation is provided in the submission
- **It's possible to trace requirements and follow the process.**
  - This is included in the documentation
- **The presentation is ok.**
  - Hopefully!

Grade 4 (G) (NOTE: This is a list of possibilities, not the union of ...)

- **A somewhat larger application with a more sophisticated design,**
  - The program includes 4.8k source lines and several included design pattern
- **Possibly use of (needed) design patterns.**
  - The program uses several different design patterns, including but not limited to:
    - Singleton
    - MVC
    - Observer
    - State
- **Solid code and packaging, everything is easy to locate.**
  - Packaging and subpackaging based upon MVC pattern, with additional subsystems.
- **Clean subsystems.**
  - Subsystems are indendent and with clear clean responsibilities

- **More functionality (use cases) implemented, possibly attention to nonfunctional requirements.**
    - The application includes a total of 13 use cases. Several non-functional requirements are considered.
- **Good control over dependencies, clean interfaces.**
    - Hard to define exactly. Strictly kept to MVC pattern, and tried to code towards interfaces as much as reasonably possible.
- **Possibly use of external libraries.**
    - The application uses libGDX as an external library for handling graphics
- **A more advanced GUI.**
    - We've put a lot of thought into the GUI, and except the resolution, it mostly resembles the initial mockup
- **External configuration and data.**
    - Data is externally gathered via an AssetManager
- **Test suites cover a lot of application code.**
    - Test suites cover 100% of the model.
- **Documentation is short and correct and obviously useful for others. The presentation gives a good view of the strength and weakness of the application.**

Grade 5 (VG)

- **Like grade 4 but even more and with higher technical level and obviously smart features**
    - While "smart featuers" is complex to define, here are some examples:
        - Using an AssetManager to easily handle the usage of textures, sound files, and
        - Using a LevelBuilder to parse textfiles into levels, allowing levels to be easily created with just an image and a .txt file
- **Possibly some kind of modular design with plugins, or other advanced design.**
    - As much as reasonably possible, we have tried to keep with good modular design staples, towards interfaces and not concrete implementations, and with distinct separation of concern.