This year ICS4U will be creating Let’s Make Pizza, a board game created by Family Pastimes an Ontario Company that’s been in business for over 25 years.

As a class, two apps, and possibly a packaged PC game, and some raw code will be created and sent to the owner. The students are not responsible for maintenance following the delivery of the game.

The students are allowed to keep their copy, but may not distribute it in any way.

Minimum Documents Required:

* Daily personal logs
* Gantt chart(s)
* Flow chart(s)
* Logs appropriate to their role in the project
* Final report
* Specifications
* Any documentation related to the game we produce required by the Owner of the game, Mr. J Deacove.
* Thank you letter at the end.
* Possibly: school picture or other of students added to the credits

Roles:

# Project Manager

Curriculum Specifically: B 1.3, B1.4, B1.5, A4.4, C.21

# Tester

Curriculum Specifically: A4..2, C2.1 – C2.4

# Programmer(s)

Curriculum Specifically: A4.3, A4.4, C1.1 – C1.4

# Communications & System Analyst (Document Expert)

Curriculum Specifically: B1.1, A4.1, C2.1 – C2.5

# GUI developer

Curriculum Specifically: Curriculum Specifically: B 1.3, B1.4, B1.5, A4.4, C.21

# All

Curriculum Specifically: B1.2, B1.6, B1.7, B2.1, B2.2, B2.3

# You will each rotate between the roles types so that everyone will code and everyone will at some point have a project manager or communication role.  I expect a personal daily log. A report/document/group or job log (gantt chart, tester log) etc. when you take over each role, excluding the first day.  I will expect the most recent log, report/document (gantt chart) etc. when you complete your role.  I will expect for some documents in the middle of the project. As roles switch the end of a role’s documents should be identical to the start of role for the next person, yet each of you should hand in the document, person A as end and person B as beginning.  (Yes I’ll have duplication, but they will be filed differently under two different students and will be checked for consistency or inconsistency if need be.  Interpretation sometimes does play a part particularly in the testing role.)

You will be performing each type of role during the project.  Here is the plan in theory

Day 1 & 2 til 3: (Stage 1 – Stage 2 begun?) – can rush other pieces, but don’t rush this stage, be quick and effective.

1. Decide initial roles
2. Create initial flowchart
3. Decide on your form of log, report, communication
4. Stages, Milestones etc. should all be determined and implementation begun – everything logged
5. Hand in these details – will be a folder for each stage in Google Classroom**.**

Project layout:

1. Your **comments** at the top of your work/flow chart or program, should contain the following:

* The Game Name.
* Your Name and date you began creation/coding (at the top for coding, clear on flowcharts too.)
* Modifications you made, with dates
* A MUST, all work, flowcharts, the Code pieces you borrowed, and ideas must be properly acknowledged (Comments in your code, footnotes on other pieces). You need to state, from where you borrowed, book/page/teacher/class notes with date/friend etc. and the link (You are encouraged to borrow “good” code and acknowledge the author/creator/writer). We borrow code all the time in the field, and we acknowledge the author. Failure to properly acknowledge what is borrowed is plagiarism on Communications and possibly the entire question. NOTE: You may not borrow more than 25% of your code. Ideally, we should be borrow as little as possible.
* White space to make all of this easy to read
* Any notes or cautions about quirky things or test data.

1. Splash screen or opening introducing the game (include the company that created the original board game)
2. The game
3. Help menu and instruction screen
4. Closing screen /game over
5. Credits
6. About ? 

Program requirements

1. Modular program(s) Swift and Java with multiple classes, functions, etc.
2. User-friendly I/O and messages
3. Appropriate data types
4. Conditional structures, loop structures,
5. Custom methods
6. Arrays
7. Searching & sorting as required
8. External file I/O
9. User names (password option?)
10. GUI and enhancements, win/lose animation, graphics

Likely Stages combined for flow:

# Stage 1: (Analysis and some Design)

* Roles decided and for what length of time
* Flowchart
* First gantt chart
* Pert Chart
* Daily logs
* Brainstorming, determine deadlines and critical milestones
* Design
* Storyboards
* Resource list
* Project schedule
* Risk statement

# Stage 2: (Design & Develop)

* End user requirements
* Software plan
* Finish/change/tweek Stage 1 as required
* IPO charts
* Pseudo code (optional)

# Stage 3: (Implementation, Code and Document, some basic testing)

* Produce code
* Test each little class, function independently
* Record problems and solutions in project log
* Create a user guide
* Create help screens
* (see requirements section – you need all of this)

# Stage 4: Testing/Verification

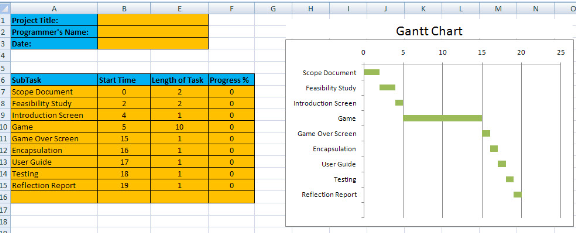
* Concurrently with stage 3, then one final big test
* Attempt to break, then fix.

# Stage 5: packaging and 2 Apps (iOS and Android) and Final Reporting

* Can run concurrently with stage 3 and 4 some of the time
* Test thoroughly, may invite community beyond out class to try Alpha version (grade 11 classes, CS Club or school in general - get buzz going for next year and unknown future)
* Final documents, reporting, user manual produced possibly in conjunction with Business?
* Reflection report and final logs handed in.
* Possibly meet the owner and hand the game to him.

**Creating a Gantt chart**

You now have the information you need to create your Gantt chart. Download the **Gantt chart template** (created for you using *Excel*) or any other one you prefer as long as it can be handed in and kept (IE: not a link).

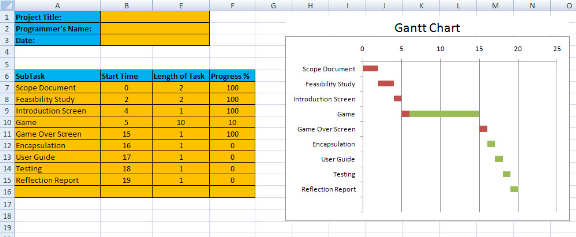


**Updating your Gantt chart**

A member of your team will need to be responsible for updating the Gantt chart on a regular basis. Submit a new chart to your teacher *every two days* until your project is complete.

This should be obvious, but when you submit an assignment to your teacher, you are to mark that task as *100% completed*.

The image below shows what the Gantt chart would look like when your group has completed the *scope document*, *feasibility study*, *introduction*, and *game over*screens, as well as *one-tenth* of the finished game.



Miscellaneous Links to help out.

## Gantt Charts

The following link shows how to convert a basic bar chart to a gantt chart. (The same can be done in Google Sheets.)

<http://www.openoffice.org/documentation/HOW_TO/spreadsheet/gantt_pm.pdf>

Another How to:

<https://www.teamgantt.com/blog/gantt-chart-template-roundup/>

(<http://www.hellotecho.com/how-to-create-a-simple-gantt-chart-with-google-sheets>

An article with examples, a little clear I believe:

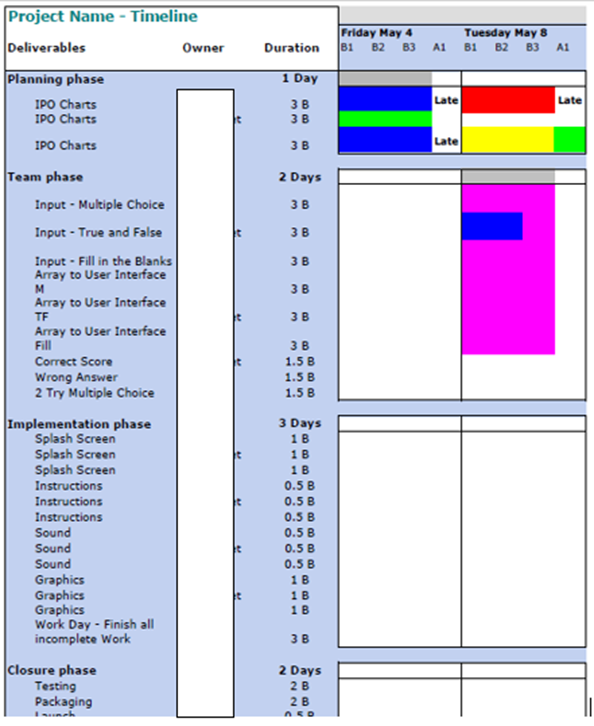
<http://www.i-programmer.info/programming/uiux/2165-whats-a-gantt-chart.html>

and a YouTube version:

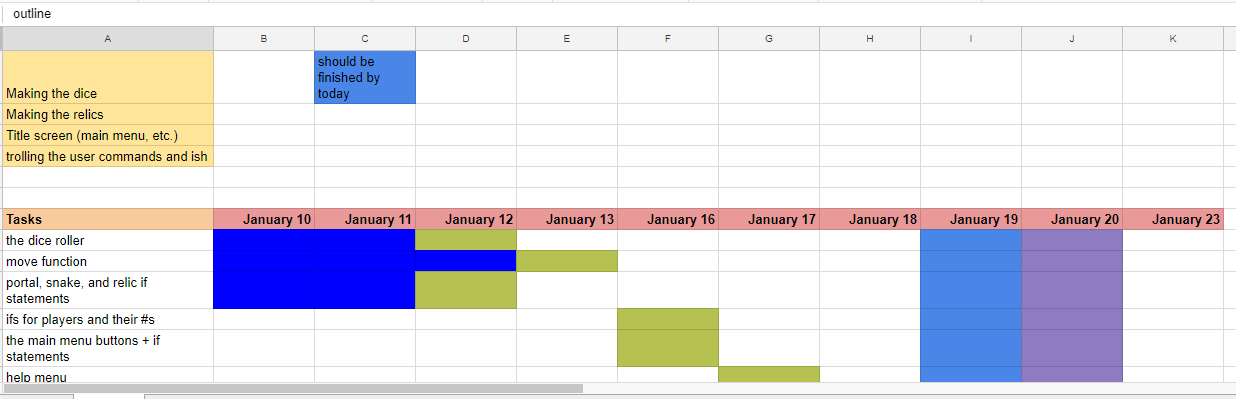
<https://www.youtube.com/watch?v=9Xc9tT6Ywuc>

There are tons of templates around – use one that works for you, imported into Excel, open office, Google sheets…

 Past example – Names have been covered.



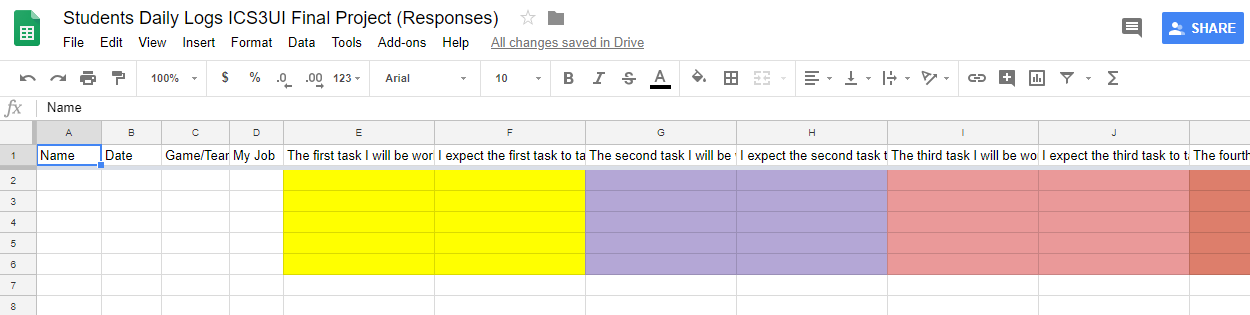
And another Example from my Students:



## Logs: (one of the one my past students used)

Here's what I need to know:

* Time - expected/estimated task (when start/when end)
* Time - actual time it took (minutes usually works best, but you could do hours with fractions .5 = 30 min.)
* The Task - specific details (Exception the first meeting and Gantt chart - enough detail would be a general estimate, specific: added all milestones and dates 5 min. Person A, B, helped.  Person C was absent.)
* single person job or task vs multiple person
* when this task is scheduled to be done/late
* (Your choice) - if using sheets/excel (each day on a different sheet OR each Role on a different sheet)  You need to hand in your log at the end of each Role so make sure you keep a copy for yourself.



Tentative Rubric (can be changed)

# Key:

Project Manager/Leader (PM),

Programmer/Coder (PC),

Tester (TS),

Communications & Systems Analyst (Document Expert) (CS)

GUI (G)

capital (A) for anything that there is a small group mark for.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Level 4** | **Level 3** | **Level 2** | **< 50 Below all expectations.** |
| Problem solving characteristics (t)  (A) | The game presents well-ordered, scaffolded solution. | The game presents well-ordered, scaffolded solution with one or two minor errors. | The game presents well-ordered, solution with one major flaw overlooked | Less than level 2. |
| Ease of Use (k) (A) | The game is easy to understand and requires minimal time for user to become proficient. | The game may require practice time for user to become comfortable with the game. | The game requires a substantial amount of instruction in order for user to be successful. Users may require special rules in order to understand the game controls and play. | Less than level 2. |
| Collaboration at beginning Logs/flowchart, help with original gantt chart (A) – handed in after first meeting. Gantt chart will change daily. | The game design allows team to solve problems or tasks throughout all stages. | The game design allows several team members to solve problems or tasks throughout all stages. | The game design allows collaboration between only two members at a time. Collaboration opportunities may be restricted at some stages (reasons should be in log). | The game design allows minimal collaboration between members. Collaboration opportunities may be restricted at some stages or single members ‘took over’ producing a solutions they wanted buy in for. (reasons should be in log). |
| **Documentation – 10% (T)**  Did student(s) include citations for sources and permissions for non-student produced materials? | ALL required permissions present OR none needed. | Most required permissions present. | Some required permissions present. | None of the required documentation present. |
| Overall (A)  (location of Schools will be taken into consideration for teams across schools – ie: two school not in close proximity will be graded more easily on this item.)  5% | Project works correctly, no major failures, no minor failures and collaboration excellent. | Project works correctly, no major failures, possibly a minor failure and collaboration excellent. | Project works correctly, no major failures, a few minor failures and poor collaboration. | Project needs improvement, (see level 2, anything less). |
| Specifications (PM) Stage 1 and 2. PM takes lead. | Problem is clearly defined. Specification is complete and appropriately detailed. | Problem is defined. Specification is mostly complete, but perhaps not entirely appropriately detailed. | Problem definition is deficient in some way, or specification does not adequately represent the problem. | Less than level 2. |
| Gantt chart and documentation, team leadership. (a) (PM) – specific to your role. | Leader is able to answer specific questions about their project and the decisions and tools used to program and design the game. (The Why’s are all answered and stated.) | PM/leader is unable to answer minor questions about the project or the decisions made. PM needs to inquire/check on  specific aspects of the project and/or design process and get back. | PM/leader is unable to answer minor questions about the project or the decisions made. PM is unclear about specific aspects of the project and/or design process. | Leader/PM did not do their job properly. |
| Communication of schedule including dates communicated across team AND with teacher. (PM lead, all group members agreed.) Stage 3 til end. Shared Google Doc? (PM) (c) | Excellent, all team members and your teacher know the status. | Good, all team members and your teacher usually know the status. | Adequate, all team members and your teacher know the status within 2 – 3 days. |  |
| Project Management (finishing) (PM) (c) | Project is an interactive, finished game with a clear rule set and goal in mind. Game is playable by one or more people. (or at least to the point it should be where you ended your role.) | Project is not completely functional or lacks a clear rule set or goal. Game may not even be fully playable. |  | Project does not work at all. |
| Time Management (t & a) (PM) at end. | Assignment turned in on time and neatly with all sections clearly labeled. | Assignment turned in neatly with all sections clearly labeled. Assignment up to 24 hours late but otherwise turned in correctly. | Assignment turned in neatly with some sections not clearly labeled. Assignment up to 48 or more hours late but within the week. | (see level 2, anything less).  This will seriously affect your final project, you need to produce a report indicating the issues and how you plan to avoid this on your final project which will have begun. |
| User experience (t & c) (TS & CG) | Program handles erroneous or unexpected input gracefully; action is taken without surprising the user. | All obvious error conditions are checked for and appropriate action is taken | Some obvious error conditions are checked for and some sort of action is taken. |  |
| GUI – (a)(CG) | Project displays a completed GUI in its entirety. The project has a clear beginning, middle, and end. | Project is complete and does not function as originally intended, but has a clear beginning, middle , and end. | Project is incomplete or does not project clear beginning, middle and end. | (see level 2, anything less) |
| GUI User interface (c) (CG) | Screen based instructions and final output are clear, correct and attractive. Program is “user friendly” with informative and consistent prompts and messages. | Screen based instructions and final output are mostly clear, correct and attractive. Program is “user friendly” with informative and consistent prompts and messages. | Screen based instructions and final output are not clear, are not correct or are not attractive. And/or Program is not “user friendly. |  |
| Communication – comments (CG) | Initial comments are complete. Internal documentation is complete and well suited to the program | Initial comments are complete but internal documentation is in some small fashion inadequate | Initial comments are incomplete or internal documentation is inadequate. |  |
| Tester – Debugging (a)(TS) | Debugging included in logs with additional programming techniques, data and advice to team for fixing. | Debugging included in logs with additional programming techniques, data and advice to team for fixing inadequate or flawed. | Debugging included in logs with additional programming techniques, data and advice to team for fixing not included. | Debugging needs improvement, (see level 2, anything less). |
| Test Cases (a) (TS) | Clear and well thought out test cases presented that cover all boundary conditions and a comprehensive range of user inputs. | Good test cases, but some boundary conditions are missing. Range of user input mostly well thought out. | Little in the way of test cases. One or more obvious boundary conditions missing. | Test cases need improvement, (see level 2, anything less). |
| Modularization & general Coding (PR & GC occasionally) | Program broken into well thought out elements that are of an appropriate length, scope and independence. Individual elements are written in a way that actively invites reuse in other projects. | Code elements are generally well planned and executed. Some code is repeated that should be encapsulated. Individual elements are often, but not always, written in a way that invites code reuse. | Code elements exist, but are not well thought out, are used in a somewhat arbitrary fashion, or do not improve program clarity. Elements are seldom written in a way that invites code reuse. |  |