Dungeons and Dynamite – Project Requirements

TEAM MAGIC DYNAMITE

MAXWELL CRAWFORD, RONALD CHAPLIN, SHANADE BEHARRY

2017

**Table of Contents**

1. Introduction
   1. Description
   2. Requirements Gathering Methods
2. Requirements Specification and Scope
   1. General Requirements (“what is needed of the system?”)
   2. User-Computer Interface
   3. System Interfaces
   4. Prioritization
3. Models
   1. Scenarios (Descriptions of the Use Cases)
   2. Class Diagrams
      1. Player
      2. Dungeon Master
      3. Report Generator
   3. Activity + Data Flow Diagrams
   4. Entity Relationship Diagram
      1. Player
      2. Dungeon Master
      3. Report Generator
   5. Use Case Diagrams
4. Plan
   1. Estimates
      1. Cost
      2. Time
      3. Team needed
   2. Risk Management Plan
   3. Task Planning
      1. Work Breakdown Structure
      2. GANTT Chart
      3. PERT/CPM
5. Meeting Notes

Introduction

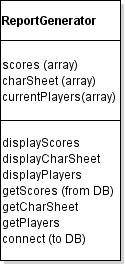
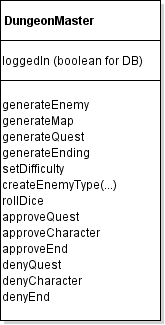
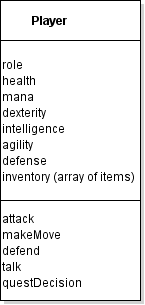
* Description:
  + This project is Dungeons and Dynamite, and it combines a role-playing text-based game with a central database, a Report Generator module, and an Android app, which will give game and information access to “Players” and to the administrator, known as the “Dungeon Master”. The Dungeon Master creates the game-board and sets up variables such as quests and difficulty, and approves the completion of the game and its objectives. Players will use the app to gather information on the game and their current quests, make moves, and create their character. The Report Generator module will be the interface between the database and the app, gathering data and presenting the information to the users.
  + We made this because we believe the Dungeons and Dynamite game needs:
    - Extra mobility, for players outside of the home
    - More flexibility over the traditional board-game format
    - Automation of roles, such as rolling the game dice, creating specifications for parts of the game, and administrative dungeon master roles
  + What will this accomplish?
    - Fun and enjoyment
    - Automation of potentially monotonous roles in the game
    - Modernization of a classic tabletop-style game
    - Player synergy and social connectivity
* Requirements Gathering Methods:
  + Group face-to-face meetings
  + Email
  + Google Drive (cloud collaboration and file sharing)
  + Slack, online chat site for the group

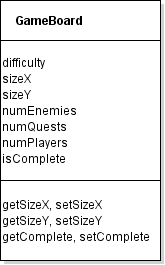
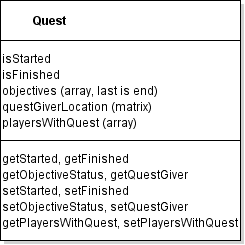
Requirements Specification and Scope

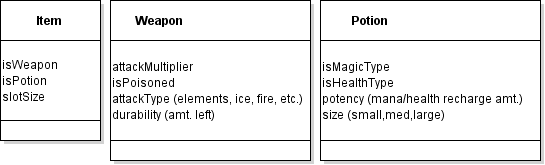
* General Requirements:
  + This system needs to have a mobile app component for player and administrator interaction with the game, a module which can interact with a database and present information to the users and administrator, and a database with tables for game characters (per player) and game information (manipulated by the administrator). All of this will be within the context of a Dungeons and Dynamite role-playing game theme, but built from the ground up for modern portability. Lastly, the system will bring some automation to the administrative roles performed by the Dungeon Master, such as dice rolls, approvals, and modifications to the game parameters.
* User-Computer Interface:
  + Mobile app
    - Inputs:
      * Player: login/registration (into database), game interactions (create a character, make moves, perform/accept quests, etc.), game completion (to be overseen by Dungeon Master)
      * Dungeon Master: login/registration, registration of a game (must do this before players can join), generate game parameters, approve quest completion, approve game completion, generate game ending, closing of a game
    - Outputs:
      * Report Generator module: show player statistics (from their characters in game), show overall game statistics (from all players), generate report of top statistics
  + Desktop app (separate instance of Report Generator)
    - Inputs:
      * Player: login to instance of existing game
      * Dungeon Master: login to instance of existing game
    - Outputs:
      * Report Generator module: show player statistics (from their characters in game), show overall game statistics (from all players), generate report of top statistics
* System Interfaces:
  + Mobile app, Android platform (Java code)
  + Desktop port of Report Generator module, view game data only
  + Database, interacted upon by mobile app and desktop Report Generator queries
* Prioritization:
  + Interface between mobile/desktop applications with the central database, and maintaining concurrency between many players (only allow one Dungeon Master to be logged in at once)
  + Maintaining a friendly, easy-to-use graphical user interface on the mobile application, so that all kinds of users/players can enjoy it

Models

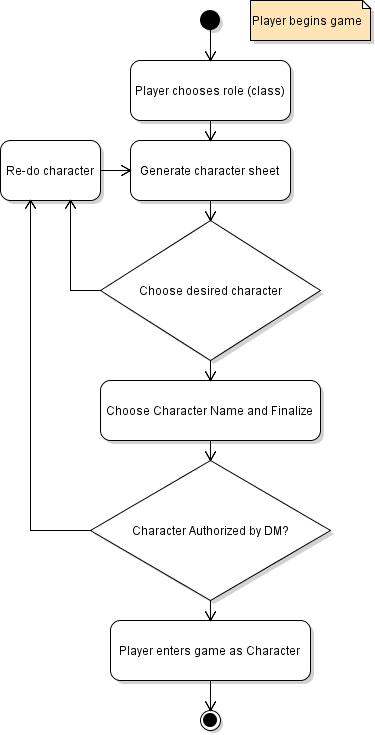
* Scenarios (Use Case Descriptions):
  + Players start a game, which includes creating a character within a game managed by the Dungeon Master, who will approve the character’s entry into the game. Once all characters are created and registered into the game, the Dungeon Master approves and initiates the game, uploading all data into the central database, with tables for each player in the game.
  + Players play a game, which includes moving their character, talking to NPCs (non-playable characters within the game world), accepting quests, and trying to finish quests by performing objectives. When those objectives are completed, the Dungeon Master may approve the completion of the quests, with an option for the Dungeon Master to generate more quests.
  + Player finishes the game, which includes player turning in their last quest and clearing their quest log. The Dungeon Master will check if there are any more quests needed to complete the game, if not, they will generate an ending (which may be positive, negative, or even neutral) for the game. The game will then be completed and updated in the database.
  + Player requests information from the mobile app, which could include scores, current quest log and objectives, and their current inventory on their character. The Report Generator module will download data from the relevant table on the database and present the information to the player/user.
* Class Diagrams:



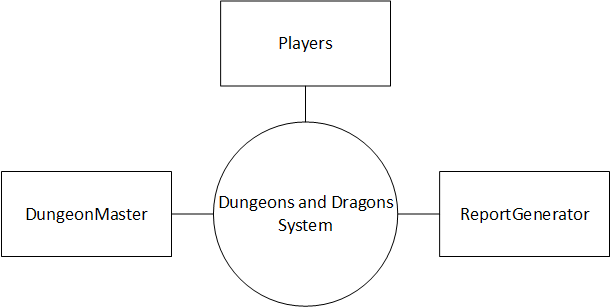




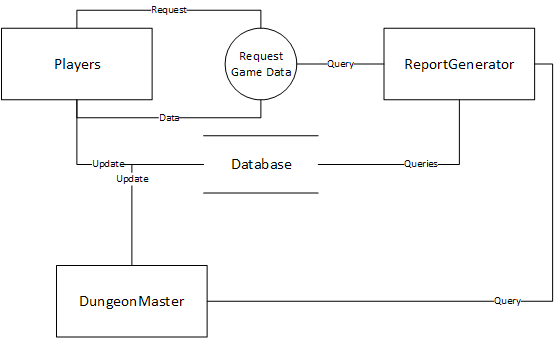
* Activity Diagrams:



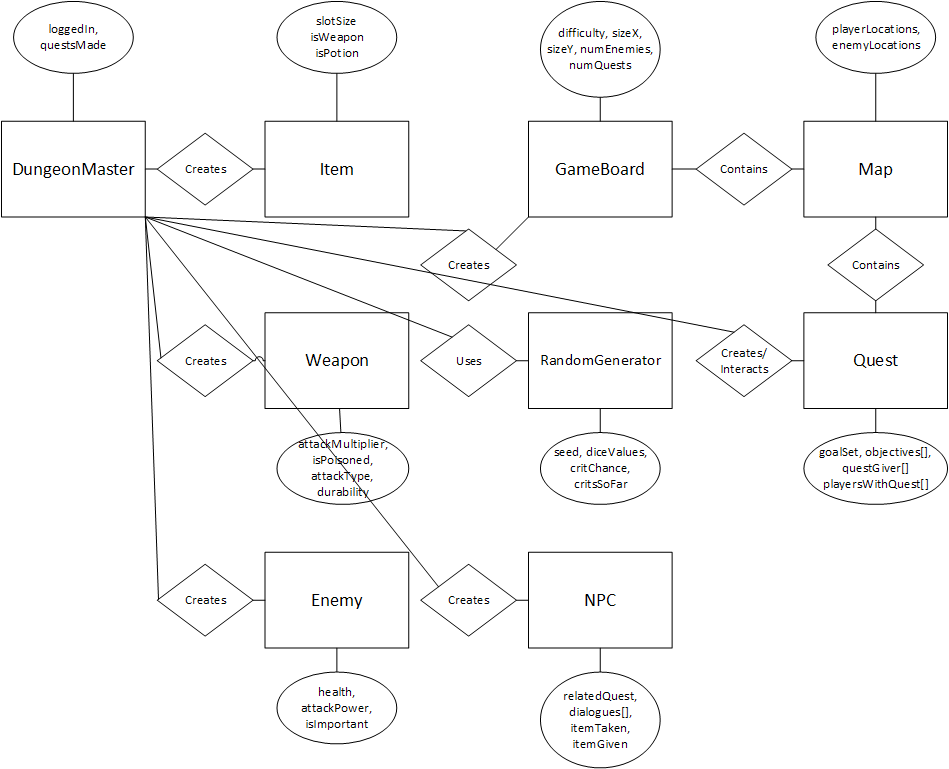
* Data Flow Diagrams:
  + **Level 0**

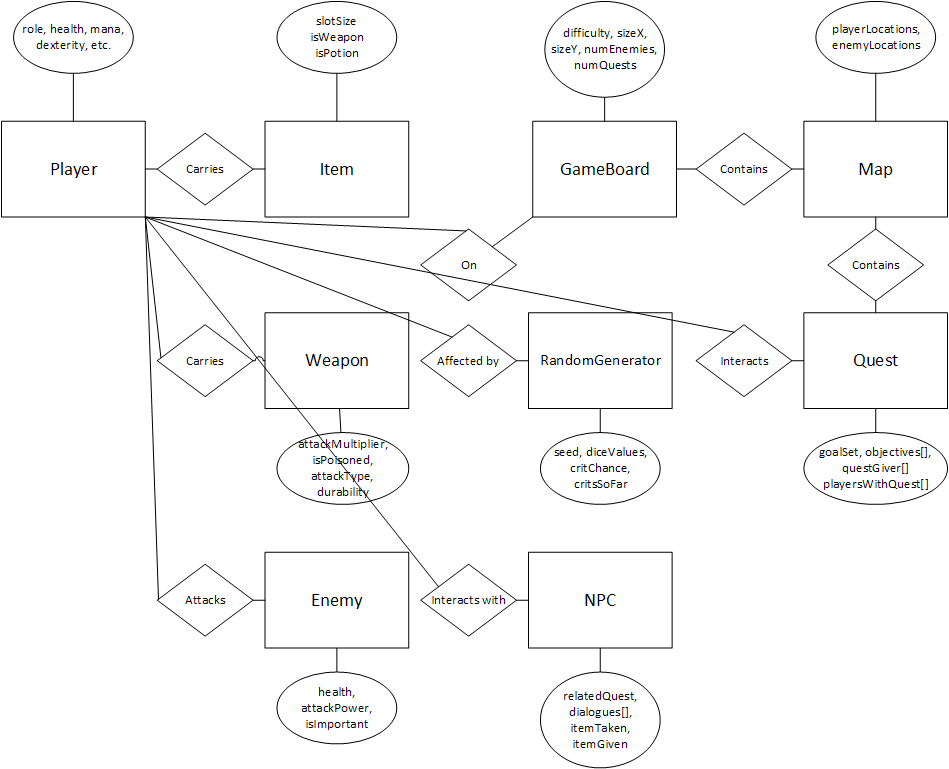


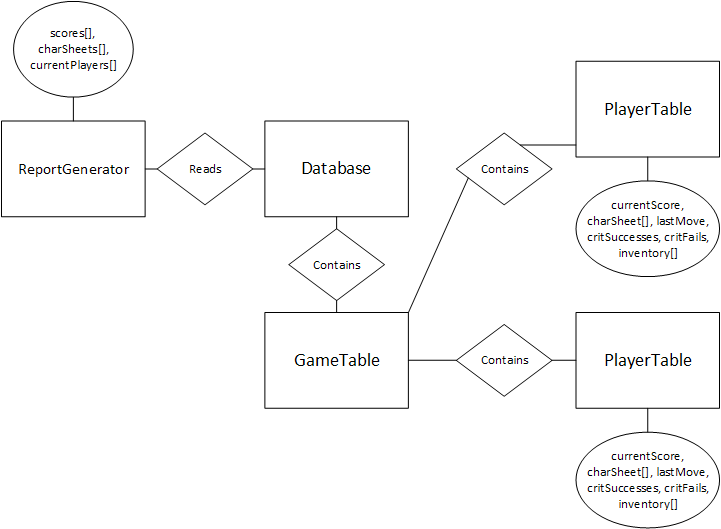
* + **Level 1**



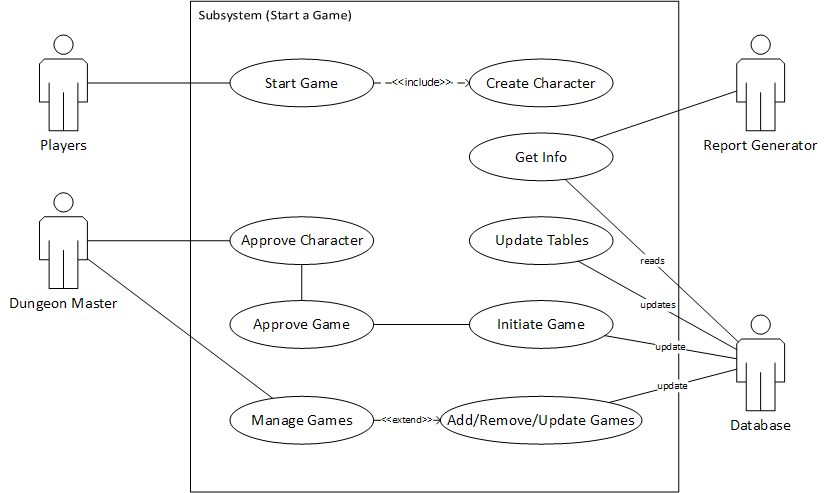
* Entity Relationship Diagrams:

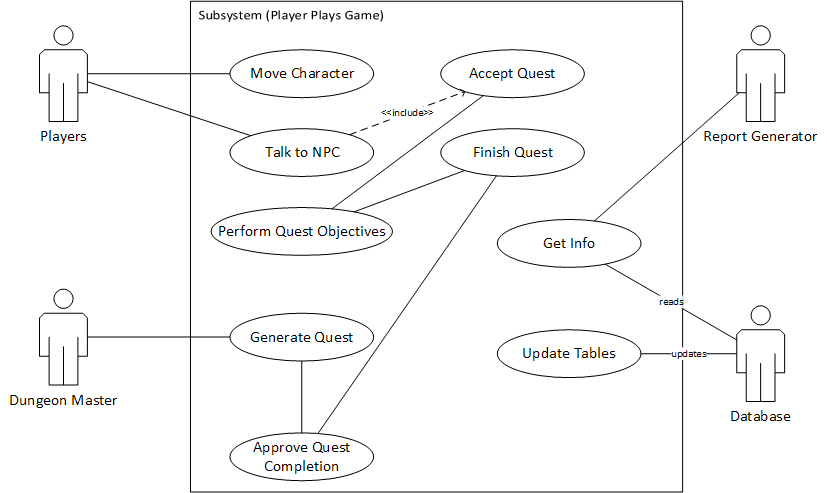


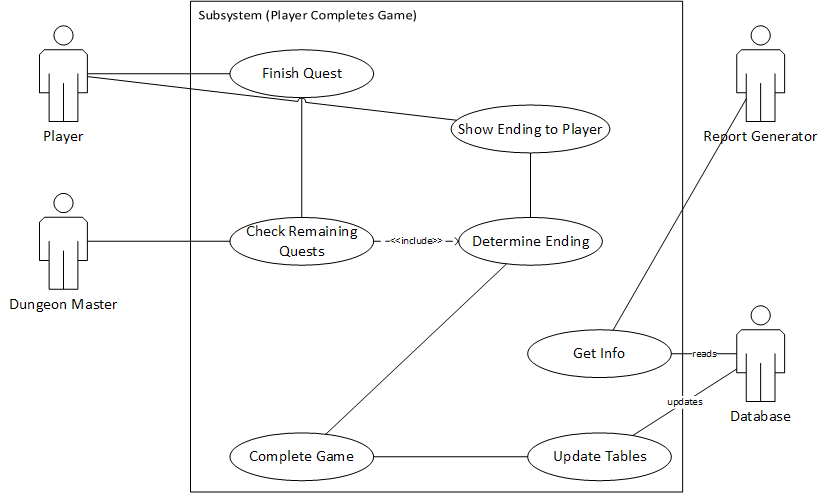


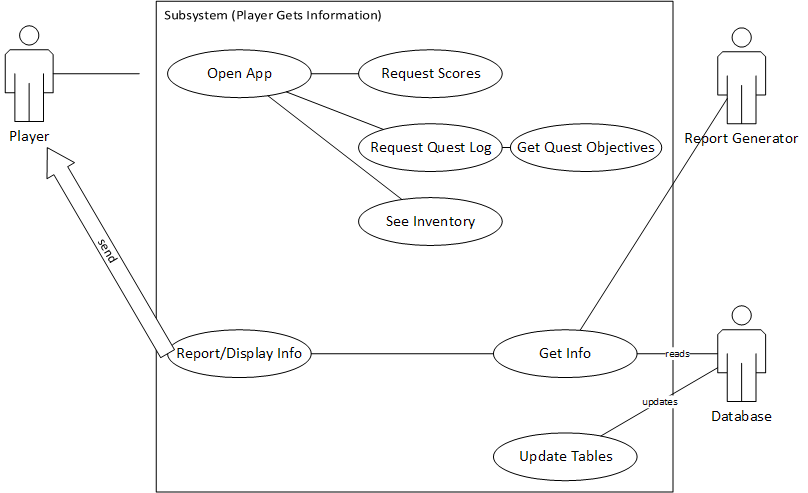


* Use Case Diagrams









Plan

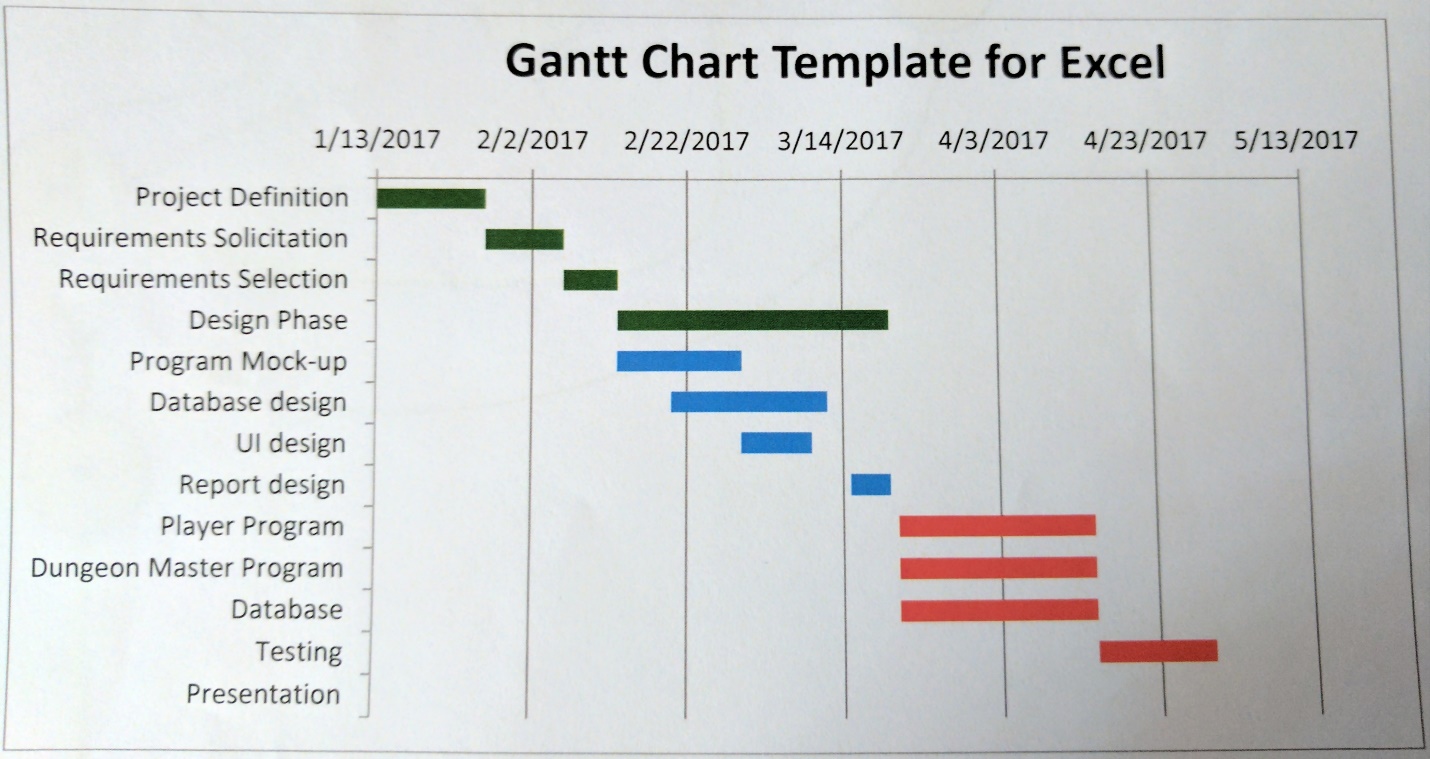
* Estimates
  + Cost:
    - $50,000 yearly salary per developer, with 3 developers on the team
    - If completion time is 3.5 months, then cost will be $43,750
    - If completion time is 4 months, then cost will be $50,000
  + Time: 3.5 – 4 months, due to both constraints with other upcoming projects, and due to the combination of 3 people each putting in approximate a person-month of work.
  + Team needed: software developer (Java needed), database architect (basic SQL skills), basic systems analyst (for design)
  + Function Points: Approximately 25-30 module functions, although some additional functions may be added for user interface drawing and Android platform backend.
  + Lines of Code: Approximately 400-500 lines of code, since most of the functions will only need 20 LOC at most, and the largest function may have up to 50 LOC.
* Risk Management Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Risk | Probability | Impact | Mitigation | Monitoring | Management |
| Team member leaves | Low | Critical – Catastrophic (depends upon # of members left) | Keep all members engaged | Keep in contact with members, via email/phone/class | Delegate missing member’s task evenly across other members; drop tasks if not absolutely needed |
| Requirements change or more tasks added (Scope Creep is possible) | Medium | Critical | Design detailed and thoughtful requirements from the start | Check that every current task is relevant and that the project covers what is actually needed for the deliverable | Adapt tasks as needed; drop tasks that become irrelevant or too difficult; adapt the budget and schedule; delegate new tasks to members |
| Design (like Flowcharts/Diagrams) needs to be changed or updated | Medium | Marginal | Keep designs modular and easy to update; take time to finish designs before moving on | Check that current prototypes/implementations match the design specs, and if they deviate, see if the design itself needs to be changed | Know the reasons for the design change and only update when absolutely necessary; take time to update all effected systems |
| Program needs an additional Class/Interface/Module | High | Marginal – Critical (depends upon impact with other code) | Keep interfaces in mind when designing Classes and their data flows; plan out the code with detailed flowcharts | Check that all data flows are appropriated used in the code, and that code coverage is as high as possible | Implement the needed module while communicating with team members about other potential systems/modules that could be affected |
| Program fails a Test Case during Test Phase | High | Negligible – Critical (depends upon Case importance) | Write code with test cases in mind; predict how the data will be used and how the user could use the interfaces | Continue to test software as we implement modules, rather than testing at the end of all modules | Test all affected systems of the failing module, and debug the error path; have many members evaluate the module |
| Lack of proper communication | Low | Marginal | Use all available communication mediums and make sure every member knows how to stay in touch | Never allow long gaps of time without regular communication; have a mix of long and short burst chats | Call a face-to-face meeting when long gap occurs, and re-establish communication methods |
| Lack of Task Delegation and Management | Medium | Marginal | Keep and update a list of Tasks and a schedule for them (alongside budgets) | Continually communicate with members and get progress reports | Confront members who are behind on tasks; go through task checklist and assign missing tasks |
| Lack of Team Motivation | High | Critical – Catastrophic | Continually communicate with members and allow “down-time” in between tasks | Communicate with members and assess morale, both overall and individually | Directly address concerns with affected members, and be able to assign bite-sized tasks that are easier to complete |

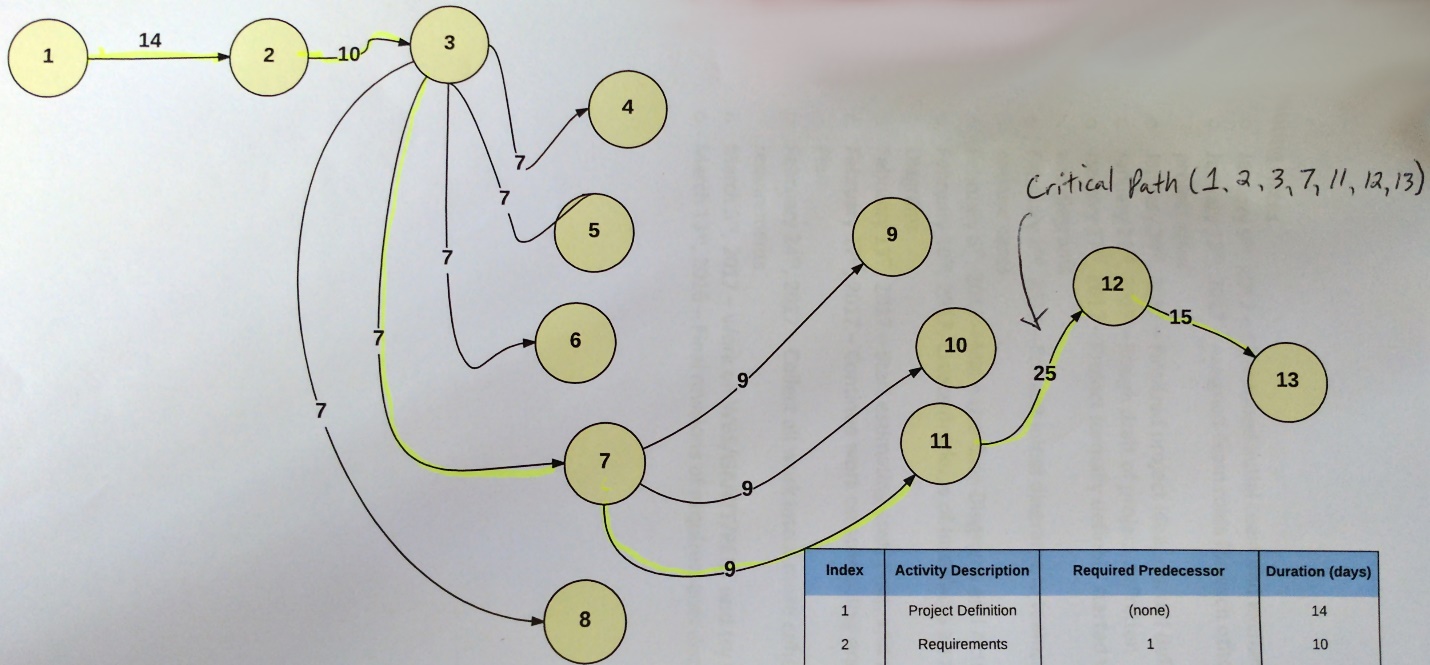
* Task Planning
  + Work Breakdown Structure

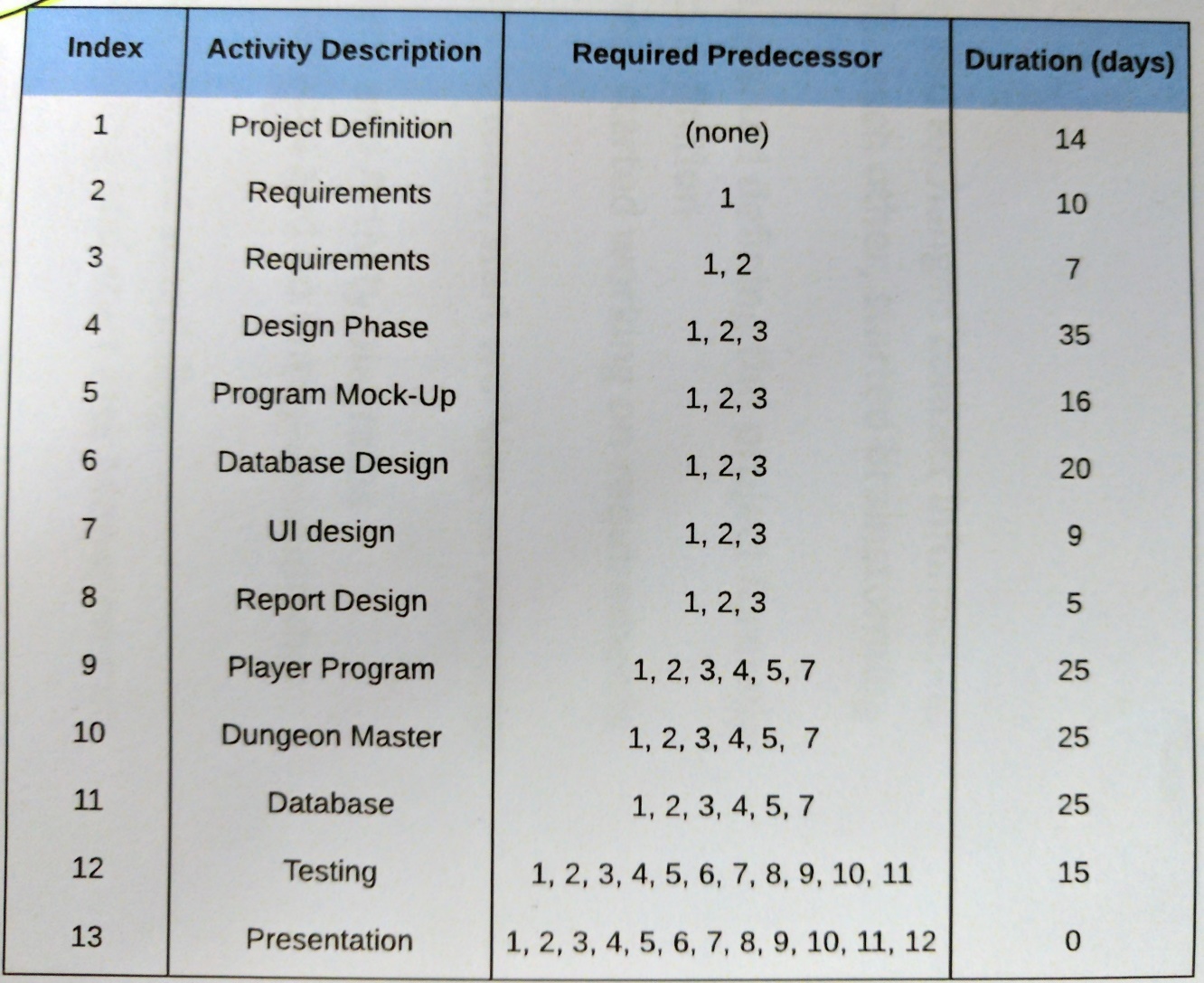
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task # | Task Name | Start | End | Duration (days) | Dependencies |
|  |  |  |  |  |  |
| 1 | Project Definition | 1/13/2017 | 1/27/2017 | 14 |  |
| 2 | Requirements Solicitation | 1/27/2017 | 2/6/2017 | 10 | 1 |
| 3 | Requirements Selection | 2/6/2017 | 2/13/2017 | 7 | 1,2 |
| 4 | Design Phase | 2/13/2017 | 3/20/2017 | 35 | 1,2,3 |
| 5 | Program Mock-up | 2/13/2017 | 3/1/2017 | 16 | 1,2,3 |
| 6 | Database design | 2/20/2017 | 3/12/2017 | 20 | 1,2,3 |
| 7 | UI design | 3/1/2017 | 3/10/2017 | 9 | 1,2,3 |
| 8 | Report design | 3/15/2017 | 3/20/2017 | 5 | 1,2,3 |
| 9 | Player Program | 3/21/2017 | 4/15/2017 | 25 | 1,2,3,4,5,7 |
| 10 | Dungeon Master Program | 3/21/2017 | 4/15/2017 | 25 | 1,2,3,4,5,7 |
| 11 | Database | 3/21/2017 | 4/15/2017 | 25 | 1,2,3,4,5,7 |
| 12 | Testing | 4/15/2017 | 4/30/2017 | 15 | 1,2,3,4,5,6,7,8,9,10,11 |
| 13 | Presentation | 5/1/2017 | 5/1/2017 | 0 | 1,2,3,4,5,6,7,8,9,10,11,12 |

* + GANTT Chart



* + PERT/CPM





* Meeting Notes
  + January 9th, 2017 – Established initial team and exchanged contact information
  + January 13th, 2017 – Assigned team roles for each other, started brainstorming project ideas
  + January 20th, 2017 – Finalized project idea, started defining the project formally
  + January 23rd, 2017 – Rough draft of project definition
  + January 27th, 2017 – Project formally defined, started working on requirements and diagrams
  + February 2nd, 2017 – Present initial diagrams to team, start working on scenarios and use cases
  + February 6th, 2017 – Make Use Case Diagrams and Activity Diagrams
  + February 10th, 2017- Create mockups of Interfaces and do Entity-Relationship Diagrams
  + February 13th, 2017 – Start estimating cost, time, and scheduling
  + February 17th, 2017 – Continue work on scheduling and start Risk Management Plan
  + February 24th, 2017 – Collect all work into a more cohesive document and revise requirements
  + March 3rd, 2017 – Work on WBS/GANTT/PERT and try to establish critical path
  + March 13th, 2016 – Final revisions of requirements documents and diagrams