SOFE 3490: Software Project Management

Lab 3

Course Project: **Music Hub**

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| Umar Ehsan | 100634240 |
| Diba Shojaeigoradel | 100621768 |
| Ryuji Komai | 100631883 |

**Project: Music distribution**

**1.0 Project Estimation:**

The estimated effort and duration for the music hub project is calculated using the COCOMO model. For the purposes of the model, we identify our project as an Organic project due to the following:

* The project domain consists of previous works which are very similar.
* Knowledge of development techniques is well understood with team members having nominal experience with the technology needed.
* The project team is small and thus requires less complexity.

In order to calculate the project effort estimates we use the basic COCOMO model as described below.

The basic COCOMO model calculates project effort in person-months using the following formula:

E = a (KLOC) ­­b where E is effort calculated in person months

D = c (E) d where D is duration calculated in months

Where KLOC is the thousand lines of code and a, b, c, d are constants.

For an organic project, a = 2.4, b = 1.05, c = 2.5, d = 0.38

Since the project will involve both a web-app and a mobile app, and there are previous solved problems with libraries available for this problem, the estimated lines of new code is around 1500 (web) + 500 (mobile) = 2000 lines of code.

Hence E = 2.4 \* (2)1.05 = 5 person months

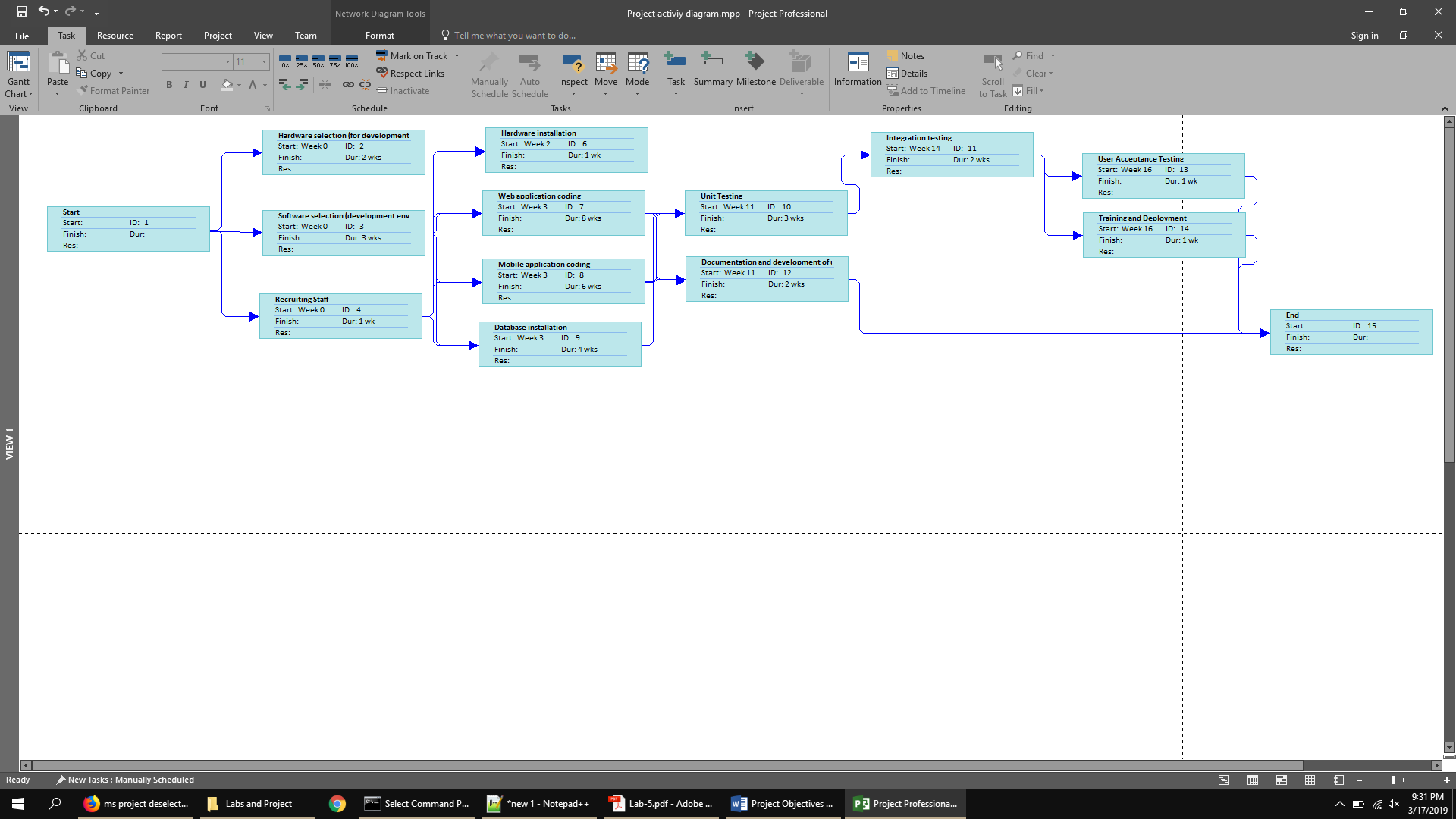
And D = 2.5 \* (5) 0.38 = 4.5 months

Hence the project is estimated to take 4.5 months with an effort of 5 person months.

**2.0 Project Activity Diagram:**

The project activity diagram along with an activities’ chart is shown below.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Duration** | **Earliest Start** |
| Hardware selection (for development and server deployments) | 2 weeks | Week 0 |
| Software selection (development environment, libraries, testing framework) | 3 weeks | Week 0 |
| Recruiting Staff | 1 week | Week 0 |
| Hardware installation | 1 week | Week 2 |
| Web application coding | 8 weeks | Week 3 |
| Mobile application coding | 6 weeks | Week 3 |
| Database installation | 4 weeks | Week 3 |
| Unit testing | 3 weeks | Week 11 |
| Integration testing | 2 weeks | Week 14 |
| Documentation and development of user support manuals | 2 weeks | Week 11 |
| User Acceptance Testing | 1 week | Week 16 |
| Training and Deployment | 1 week | Week 16 |

**3.0 Project Risks and Mitigation Strategies:**

* **Risk 1 (Software & Hardware Selection Activity):** Employees faces with issues when adopting new software tools
  + **Contingency:** On-board temporary training staff to bring developers up to speed on the tool(s) in question. If available and supported, the cheapest and most thorough option may be support staff from the producers of the tool, if unavailable, hire from reliable sources. Dedicated training day(s) may add up to an extra week to our scheduled release.
* **Risk 2: Moral / Burnout, especially during critical work periods**
  + **Contingency:** More preventive than reactive, ensure cohesion in the workplace on an employee level, as well as emphasize mutual respect between all inhabitants of workplace. Avoid late or weekend beta-releases to ensure developer’s time off is actual time off and not filled with anxiety/stress/on-call tasks.
* **Risk 3: Scheduled & Unscheduled Server shutdown post-launch**
  + **Contingency:** In the case of a scheduled server shutdown, allow users 24 hours of notice. This app is not critical in its functionality, thus a small period of notice allows time for music downloads and such before the shutdown. Keep downtime short by planning out the maintenance/ changes/ activities in detail. Run light regression testing to ensure no additional problems are caused. In the case of unscheduled downtime, notify all relevant developers and request immediate troubleshooting (pay off-work developers overtime). Create adequate recovery plan, implement, and emphasize heavier regression testing to ensure the fixes are not simply short-term. If a significantly complex fix is needed, short-term fixes may be allowed to give additional time to plan updates.
* **Risk 4: Customer opposition to final design/ developed prototype**
  + **Contingency:** Prepare customer feedback sessions during initial development, integration testing, and final acceptance testing. Use an agile / incremental approach to allow for rapid integration of customer feedback to minimize customer opposition to developed product.
* **Risk 5: Hardware faults**
  + **Contingency:** Develop the prototype using cloud services such as AWS to minimize downtime introduced by hardware faults during development. Arrange for procurement and installation of backup servers to account for hardware and software faults. This helps in recovering the critical application frameworks and databases.
* **Risk 6: Delayed delivery of finished product**
  + **Contingency:** Introduce a little flexibility into the schedule to allow for potential project extensions. Aim for constant monitoring of tasks accomplished and milestones to minimize the risk of delivering a late product. Avoid scope creep by keeping requirements simple and development focused around creating a robust modular design.