**Project Plan**

**Introduction**

Our client is the owner of several daycares taking care of over 600 children each day. As children grow, they need to be constantly moved to different classrooms in order to best serve different age groups. Our client would like us to create software that would better assist in the scheduling of children into appropriate age groups and classrooms, making efficient use of available resources.

Our client has a prioritized waiting list of children. A child is accepted into the daycare when the daycare can guarantee openings for at least two years. The client would like the software to manage the waiting list and give alerts when a child from the waiting list can be accepted. The primary goals of this project are to provide efficient schedules given a list of children (with varying weights / priorities), send alerts (configurable based on clients’ needs) for relevant events, and allow for manual modifications to the schedules calculated by the software.

**Project Organization**

The project is being conducted by Jonathan Brown and Keivan Sabet. Each member is represented equally. Tasks are assigned to each member through a process of collaborating on items and reaching agreement on how these items will be divided.

Currently, due to the small team size, no roles have been explicitly assigned to any group member. Work will be completed by each member as the needs arises and based on individual member’s strengths and experiences. This is subject to change at any point upon further discussion and any events as the project progresses.

**Jonathan Brown**

**Keivan Sabet**

Architect: In charge of how the software is organized and designed

Developer: In charge of writing the code for the software

Evaluator: In charge of testing and maintaining the integrity of the software

Collaborator: In charge of communicating with the client and communicating the client’s demands

Manager: Maintains schedule and documents

**Risk Analysis**

Below is a list of identified risks and the strategies for reducing their effect.

**Risk –** The Identified risk.

**Cause –** The cause of the risk

**Likelihood –** The likelihood that the risk will occur. Ranked numerically from 1 to 5 with 1 being unlikely and 5 being very likely.

**Impact –** The amount of damage cause by the risk occurring Ranked numerically from 1 to 5 with 1 being minor damage and 5 being severe damage.

**Reduction Strategy –** The strategy taken to reduce the chance of a risk occurring and to reduce the impact of a risk occurring during the project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Cause | Likelihood | Impact | Reduction Strategy |
| Underperformance | Small team size, time constraints, lack of experience | 5 | 5 | Have frequent well documented meetings. |
| Underestimate time required to complete a requirement | lack of experience, | 3 | 5 | Prioritize requirements from most important to least important. Re-evaulate time and cost to complete requirements throughout the life of the project. Focus on completing the project in deliverable increments and allow for early and frequent client feedback to ensure needs are met. |
| Client expectations of software are not met | Lack of requirements or incorrectly defined requirements | 4 | 3 | Have frequent well documented meetings with the client. Show client prototypes and receive feedback |
| Cloud service failure | Server failure, network failure | 5 | 1 | Have frequent backups of all documents and code stored in the cloud. |
| Network failure |  | 5 | 1 | Have frequent backups of all documents and code stored in the cloud. Have designated face to face meeting times and clear objectives for each member. |

**Hardware and Software Resource Requirements**

Software will be created by using personal computers to write project documentation and code. Cloud based services such as Github will be used to maintain the project code. The software being developed will be deployed on individual pcs in a test environment before finally being deployed to the client. The final release / deployment approach will be determined as the project nears completion and all requirements have been confirmed / verified.

The following software tools will be used to develop code:

Eclipse IDE (with various plugins)

Linear programming API (To be determined. Currently 2 are under investigation)

postgresql (database for testing before deployment unless >1 month trial for Microsoft Access can be negotiated).

Microsoft Access database for pre-deployment testing (client is using Microsoft Access currently. Can only setup 1 month trials, so will not be doing this until interaction with client requires this).

Documentation is written using the Microsoft Office Suite of programs which includes the use of Word, PowerPoint, and Excel.

More to be determined

**Work Breakdown Structure**

**Project Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Activity | Start Date | End Date |  |  |
| Find Client |  |  |  |  |
| Project Proposal |  |  |  |  |
| Establish Team Communication and Environmental Setup |  |  |  |  |
| Project Plan |  |  |  |  |
| Establish High-Level Requirements |  |  |  |  |
| Form Initial Prototype |  |  |  |  |
| Collaborate with Client and Refine Prototype |  |  |  |  |
| Establish Architectural Skeleton |  |  |  |  |
| Determine Class, Methods, and Tests |  |  |  |  |
| Establish Architectural Design |  |  |  |  |
| Mid-Term Presentation |  |  |  |  |
| Create Functional Skeleton |  |  |  |  |
| Code and Test High-Level Requirements |  |  |  |  |
| Create Functional Prototype |  |  |  |  |
| Collaborate with Client and Refine Prototype |  |  |  |  |
| Create Final Software |  |  |  |  |
| Final Report |  |  |  |  |

The suggested model for scheduling seems like it is more suitable for a waterfall development approach, which is not how this project is currently planned to follow. Instead the project will be completed in increments, each focusing on smaller pieces of the project and incorporating many of the phases mentioned above in each increment (specifically: class, methods, tests, functional skeleton, code and test, functional prototype, and collaboration with client, among others).

**Milestones & Core Functionalities / Deliverables**

Developing functioning linear programming model that accurately reflects the clients current data set and expectations of the software.

Database support for all relevant project data. (Database models and code to interact with database, as well as test data to reflect requirements).

Developing portion of project that can accept data and convert the linear programming model into a solvable schedule.

Create interface for the client to use that allows for client’s desired behaviors.

Allow for manual interventions to the generated schedules.

Allow for configurable alerts based on clients needs. (Need to focus especially on user-friendly configurations here, determine likely changes to be made).

More milestones and required functionality to be determined as the project progresses…

**Monitoring and Reporting Mechanisms**

We will be using Github to maintain, and expand the source code for the project.