BORG CALANEDER

MILE STONE 2

Group Memres:

Anjaneyulu Bodepudi (5973775)   
Manouchehr Azizi (5232287)   
Hermann Sonfock (5986052)   
Viet Hung Nguyen (9816240)   
Hamid Shahrestani (9729747)

**PERSONAS**

**Product Background**

In today’s hectic and stressful society it is becoming more and more common for people to have busy schedules and a lot of things to remember every day. Thus, the need for a mean to organize all this information is bigger than ever. BORG Calendar provides the system to achieve that with a highly versatile calendar which appears in every possible view (i.e. day, week, month and year views), equipped with features such as an appointment editor that enables users to add single or repeating appointments with the ability to incorporate any relevant information and enhanced with email and popup reminders, an address book that can store any kind of information (from names and addresses to phone numbers and birthdays), a to do list, memos and checklists as well as a powerful task tracking system which enables users to keep track of more complex tasks that run through various states. So it becomes quite obvious that BORG Calendar is the ideal application for anyone who wants to organize their everyday schedule in a more sufficient way. It is open source software that is written in Java, so is platform independent as long as JVM is installed on the client machine.

**Stakeholders**

* **End users/Desktop**

Who only need to have some basic knowledge in the use of computers and are most likely to use the basic features of the application, without getting involved with the more complex ones (e.g. the task/project tracking system), and the default database which requires no special setup. Of course, the program might be more useful to users with a busy daily schedule or those working on a computer quite often.

* **Advanced users**

Who have a better knowledge in the use of computers and are probably interested in exploring all of the program’s capabilities and maybe wish to set up a different type of database.

* **Developers/Testers**

Who have a deeper knowledge of programming in Java and wish to improve/extend the application and or only test the software.

* **Others**

Who wish to contribute to the project in other ways such as getting involved with the program’s documentation or translations? (This user class could be merged with the previous one)

* **Project Owner**

He is responsible to make major changes to the project. He can shut it down and stops any support for it. He can initiate a premium version of the software.

ACTORS

* Primary actors/End user

Intended to use the BORG calendar Features.

SECONDARY ACTORS

* Physical System

Service provide by the system to operate the BORG calendar

* Organization

Maintain/version controlling of the BORG calendar System

**Use Cases**

|  |
| --- |
| **Add appointment** |
| The user can create a new appointment/to-do item in the Calendar for a specific date. The appointment can occur once or more than once on a timely manner. |

|  |
| --- |
| **Select view** |
| The user can open different views in the Calendar as tabs. The available views are :  Month View, Week View, Day View, Year View, To Do List, Address Book, Tasks, Memos, Checklists, Search |

|  |
| --- |
| **Manage tasks** |
| Users can create/edit/delete tasks. Tasks are complex activities that have start and end dates, and are created in an initial state (default is OPEN) and progress through a user configurable tree of states to reach an end state (default is CLOSED). |

|  |
| --- |
| **Manage Projects** |
| Like tasks, users can create/edit/delete a project. A Project is simply a way to group related projects and tasks. A Project can set an overall due date for all child projects and tasks.  Each project can have any number of child projects and tasks. All child tasks must have due dates on or before the project due date. A project cannot be closed unless all of its child tasks are closed. |

|  |
| --- |
| **Manage Subtasks** |
| Subtasks are single units of work that do not progress through a set of states. They are either completed or not-completed. Users can create them, edit or delete them. |

|  |
| --- |
| **Manage Contacts** |
| User can add a contact to the Address Book. Address Book holds addresses and other personal information. There is a birthday field for each record in the address book. If a date is entered in this field, the person's birthday will appear on the calendar on the appropriate day each year. They can change the values for a contact later or even remove it. |

|  |
| --- |
| **Manage Memos** |
| The memo tab on the main window allows user to create and edit plain text memos. |

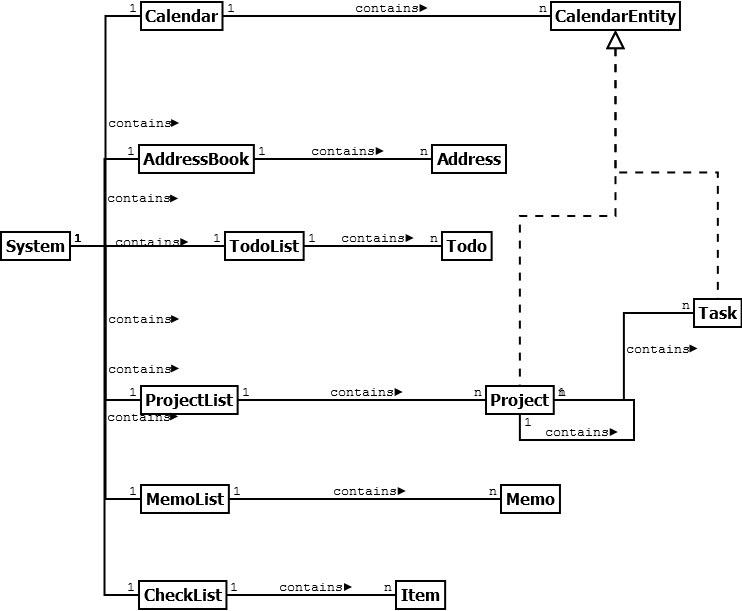
|  |
| --- |
| **Change Options** |
| Options can be changed from the ***Options*** menu. User can change the following settings:  Appearance, Fonts, Email Parameters, Popup Reminders, User color Scheme, Task Options, Encryption, Startup Views, Database Information, Miscellaneous |

|  |
| --- |
| **Search** |
| The search option from the ***Action*** menu will bring up a search window. There are a number of search criteria to choose from. User can also change the category of multiple appointments at the same time or bulk-delete several items from this view. |

|  |
| --- |
| **Print** |
| Users can select to print any of the views of the software. They can select various options for the printing. |

|  |
| --- |
| **Import/Export to XML** |
| BORG can export/import its data to/from XML. The import/export actions can be found on the month view Action menu. Import/Export provides a good way to back up data in human readable/editable form. The auto back up feature also uses an XML file to back up the data in a ZIP file. |

**UML Diagram**



**Description**

Essentally there are many classes in the original software we picked. Not all of those classes are core conceptual. Some of them are only pure fabrication and some detailed ones for UI and or so. We have included some of the important entities in our class diagram that capture the core concepts in this software. We tried to choose self descriptive names that decrease representational gap.

Most of the relationships in between entities are composition. For example A checklist contains 0-n items. There’s an inheritance relationship between