**Software Design Document**

Lab Section: Nokia Lab,PESIT

Workstation: Nokia Lab,PESIT

Date: (04/20/2013)Software Design Document

**1. INTRODUCTION**

**1.1 Purpose**

This software design document describes the architecture,system design, the human interface design of the the product, Event4u.The document details how the software requirements should be implemented and the document also provides a blueprint for the developers to follow.This document is intended for the developers and evaluators of the product, Event4u.

**1.2 Scope**

The software being developed is a web based event management centre. The product would simplify the whole process of organizing an event by

* Helping the event manager divide the co-organizers or volunteers into different teams.
* Allocating different tasks to the teams.
* Providing different views of the software depending on different roles.
* Implementing a feature to provide the event manager the ability to track the expenditure while the whole planning process is taking place.

The points mentioned above would greatly simplify the work of an event manager and would let him concentrate more on the finer aspects of the trade. The software will be designed to maximize the event manager’s productivity by providing tools to assist in automating the process of dividing the manpower into appropriate teams. The software also helps in coordinating between the event managers, team heads as well as the volunteers and supervising them, which would otherwise have to be performed manually.

**1.3 Overview**

The document is primarily intended for the developers to build the software more efficiently. The document helps in the development of code.

The first chapter, which is the Introduction section of the document, is intended to introduce the reader about the software, Event4u. The chapter deals with the reference materials, scope, definitions and acronyms which will help the reader understand this software design document in an efficient way.

The second chapter which is the System Overview, describes about the software Event4u. This chapter describes about the functionalities of the software along with the different views of software.

The third chapter deals with the System Architecture. The chapter provides the readers knowledge about the architecture of software. Development of different subsystems is discussed under this topic.

The fourth chapter, Data Design describes about the database that the software is intended to use. The chapter describes about the storage of data entities entered by the user in the software.

The fifth chapter is the Component Design which describes about the working of different components of the software.

The sixth chapter deals with the Human Overview Design which specifies about the view of user. It provides information about the user interface. It contains the screen-shots of the user interface.

The seventh chapter, Requirements Matrix uses a tabular format to show which system components satisfy each of the functional requirements from the SRS.

**1.4 Reference Material**

* SRS v2.0
* Feasibility report.

**1.5 Definitions and Acronyms**

|  |  |  |
| --- | --- | --- |
| 1 | Employer | Employer is an individual who has contacted the event organiser. |
| 2 | Event Manager | Event Manager is an individual who is responsible for the whole event and can view the entirety of the event being planned on the software.He/She is usually the lead event organiser. |
| 3 | SQLite | SQLite is software library that implements a self-contained,serverless,zero configuration, transactional SQL database engine. |
| 4 | Team Head | Team head is an individual who is responsible for all the actions undergoing under his/her team. |
| 5 | UI | UI stands for User Interface.It is defined as the space where interaction between humans and machines occurs. |
| 6 | Volunteer | Volunteer is a person who offers to take part and help in organising the event. |

**2. SYSTEM OVERVIEW**

The product, Event4u is designed to gather information about various requirements that have to be accomplished to conduct any event successfully. The software will empower the user, be it a professional or a non professional Event Manager to efficiently manage any event at hand. The software, Event4u is intended to define a development methodology for the user, beginning with the requirements phase and continuing through to the execution phase.

Every user who wants to use the software,Event4u will have to login to facilitate security and privacy. A simple UI will aid the user in easily navigating through pages and focusing on the task at hand.

There will be 4 views of the overall event:

* Event Manager
* Volunteer
* Team head
* Employer

These are some of the major functions that the software, Event4u will achieve:

* Provide different views to the user depending on his/her roles.
* Divide the available manpower into teams.
* Appoint a Team Head.
* Allocate different tasks to teams.
* Track expenditure.

An important attribute of Event4u is its independency from the kind of Operating System that the computer on which it runs, as it executes on a system independent browser. In no case is the installation of any other program required for Event4u to work. Event4u is a standalone application. Event4u does not require installation and does not modify the host system.

**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**

At the most abstract view the Todo class will be the base for every interaction related to a task in the todo list.

A task is added by a button click presented to the User which is stored in the Database.

Status is associated with every todo, either Done or Pending.

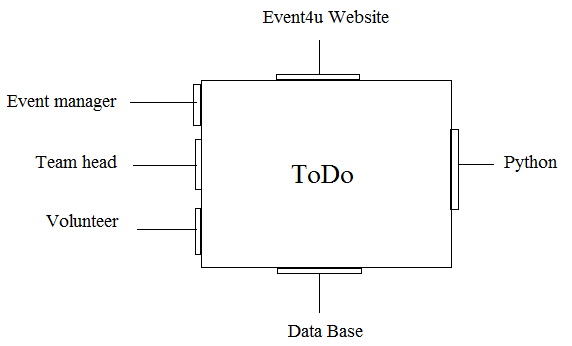


Fig 3.1(a) ACD

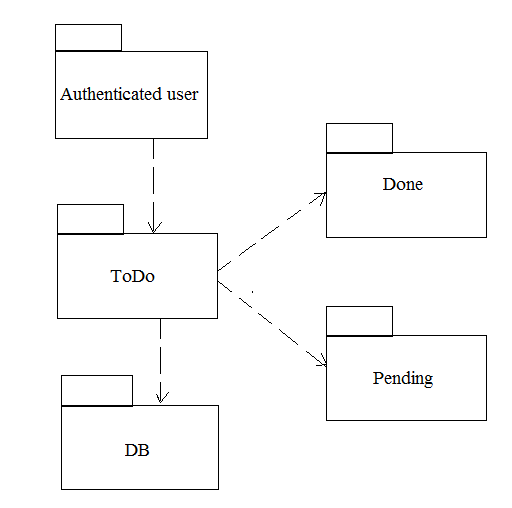


Fig 3.1(b) Archetype

**3.2 Decomposition Description**

Generalization hierarchy diagram for todo module

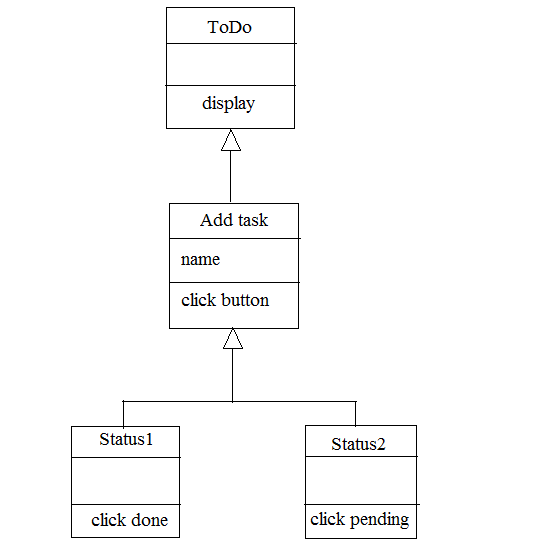


Fig 3.2(a) Generalization diagram

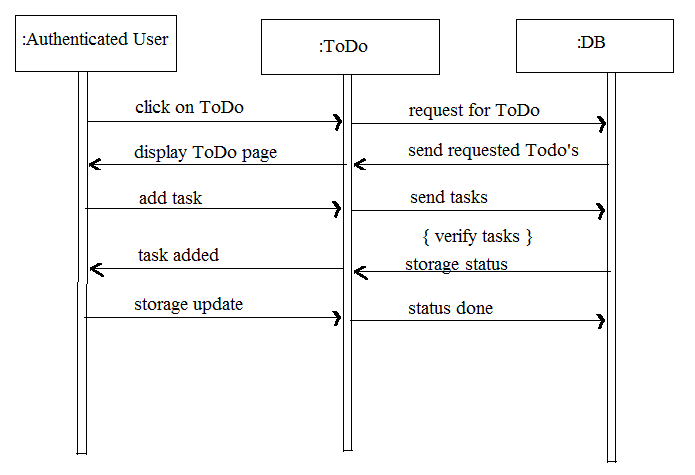


Fig 3.2(b) Sequence diagram

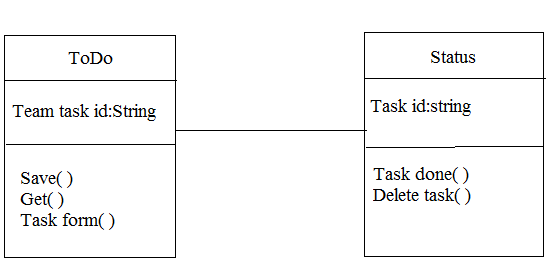


Fig 3.2(c) Class diagram

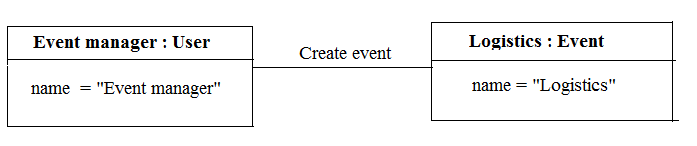


Fig 3.2(d) Object diagram

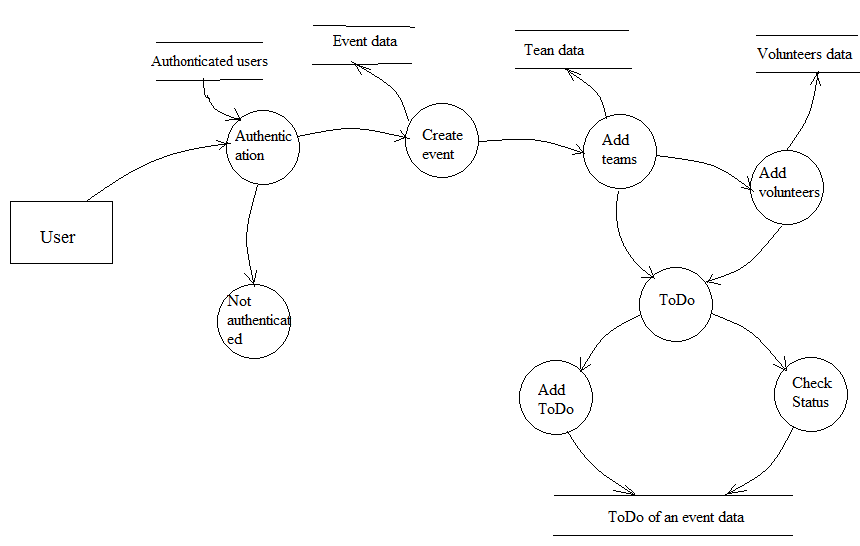


Fig 3.2(e) Data Flow Diagram

3.3 **Design Rationale**

The architecture for the software, Event4u is MTV which is known as Model Template View. MTV was chosen based on the clarity it provides for the overall software structure. The design has components with different functionalities. The todo list contains the tasks to be completed by the actors. When the task is completed, the actor should click the button done. As soon as the button is clicked, the database will be updated with the task being completed and it will be visible to the user that the task is completed.

**4. DATA DESIGN**

**4.1 Data Description**

A Todo list has been designed to store all the tasks that are to be accomplished by a team and the current status of every task.This implies that a single todo form is associated with a single team.

Thus, the todo form is accessible only to authentic users who are a part of the particular event and the particular team.

A user can dynamically add tasks to the Todo list and can mark the task as done ,thus changing the status of a particular task.

It is necessary for the changes made to the todo form to be reflected on to a Database. A todo form is unique for each team of a specific event and any changes made to the todo form should sustain when the user opens the form again. The Database used in the project, Event4u is SQLite and this maintains the attributes for the Todo list, which are as shown in the table below.

|  |  |
| --- | --- |
| **Attribute name** | **Data Type and its description** |
| task\_id | auto-incremental primary key (identifies each task uniquely) |
| team\_id | Foreign key (refers to the id of the team to which the task belongs to) |
| task\_descr | TextField (gives a brief description about the task ) |
| Status | CharField (a character depicting if the task is done or not.) |

The order of the objects to be retrieved from the database is defined in the view. View is linked to the template which provides the GUI for the Todo list.

**4.2 Data Dictionary**

Todo is a class which stores the task’s description and its status.

In order for the Todo to maintain referential integrity we have a foreign key relationship with the team ID to which the Todo belongs to.

*class Todo( ):*

*"""Todo accomplished by Teams of Event4u"""*

*task\_id = AutoField(primary\_key=True)*

*team = ForeignKey(Team, default= -1 )*

*STATS = (*

*(u'D',u'Done'),*

*(u'I',u'In progress'),*

*)*

*status =CharField(max\_length=10,choices=STATS)*

*task\_descr =CharField(max\_length=70)*

*def save(team,task\_descr):*

*””” Save the todo details into database”””*

*self.team = team*

*self.task\_descr = task\_descr*

*def get(team\_id):*

*””” Get a specific todo objec”””*

*if self.team\_id = team\_id :*

*return self*

*else*

*return NULL*

**5. COMPONENT DESIGN**

The task model object which is instantiated.

*class Task(models.Model):*

*def \_\_unicode\_\_(self):*

*return unicode(self.task\_id)*

This method describes that since each task is uniquely represented by its task ID,if there is a need to query the task,its task ID should be sent in unicode format.

A form on a web page allows the user to input data into the server for further processing.In this case, each of the todo tasks and its status has to be updated into the database .

*class TaskForm(ModelForm):*

*class Meta:*

*model = models.Task*

*exclude = ('task\_id','team',)*

This implies that the web page should allow the user to enter everything about the task except for its ID and team ID since this is assigned by the database using the primary key,foreign key relationship.

The view specifies the flow of actions to be taken in the background before the page is displayed to the user.

de*f todo(request):*

*if request.user.is\_authenticated():*

*if request.method== 'POST':*

*task\_form=TaskForm(request.POST)*

*if task\_form.is\_valid():*

*task = task\_form.save()*

*items = Task.objects.all()*

*return render\_to\_response("todo.html",{'items':items})*

*else:*

*task\_form=TaskForm()*

*return render\_to\_response('todo.html',{'task\_form':task\_form},context\_instance=RequestContext(request))*

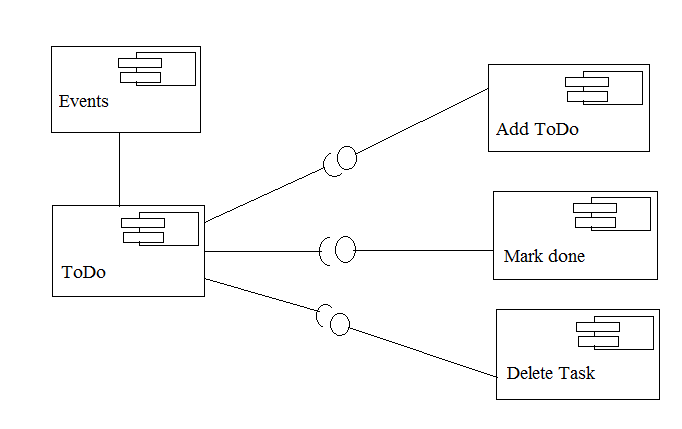


Fig 5(a) Component Design

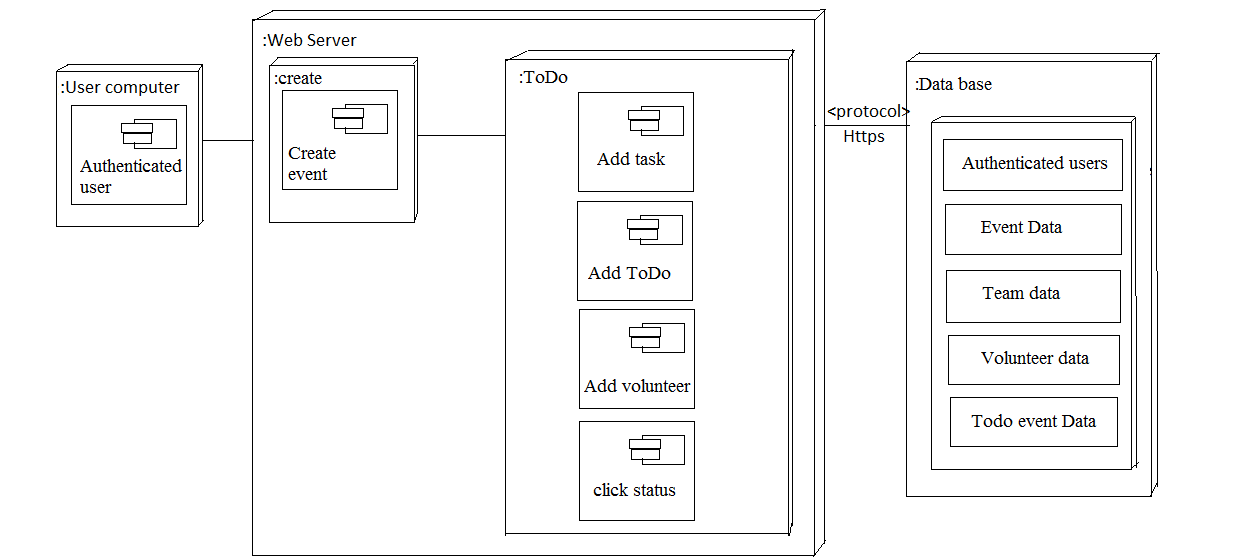


Fig 5(b) Deployment Diagram

**6. HUMAN INTERFACE DESIGN**

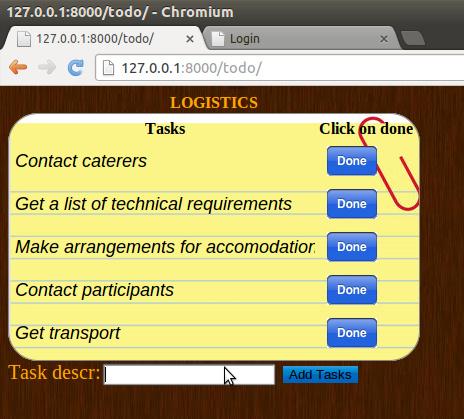
**6.1 Overview of User Interface**

The todo functionality of the Event4u helps users in checking the tasks assigned to him, marking them as done or adding new tasks into the list.

The user is given the option of selecting the event. When the user enters into the event and selects the todo functionality the user can view the tasks, add tasks to the Todo list or mark tasks as done.

If a task is marked as done, it is scratched out from the list. If a new task is added it is displayed along with the others in the same format as shown below.

**6.2 Screen Images**

l



**6.3 Screen Objects and Actions**

The user can view the tasks in a tabular form with a “Done” button placed next to each task.On clicking the “Done” button, the task is scratched out giving the client the impression of having completed a task.

A text box to dynamically add tasks is provided. Entering the task to be performed in the text box and clicking on “Add Tasks” button will add the task to the Todo list shown.

**7. REQUIREMENTS MATRIX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Components  Requirement | Allauth. | DataBase | Users. | Task. | Add Task. | Mark  Task. |
| User interface. |  | 3.1.1 | 3.1.1 |  |  |  |
| Hardware interfaces. |  |  |  |  |  |  |
| Communications interfaces. | 3.1.4 |  |  | 3.1.4 | 3.1.4 | 3.1.4 |
| Event creation. | 3.2.1 | 3.2.1 |  |  |  |  |
| Task and team management | 3.2.2 | 3.2.2 | 3.2.2 | 3.2.2 | 3.2.2 | 3.2.2 |
| Budget. | 3.2.3 |  |  |  |  |  |
| Automatic notification. |  | 3.2.4 |  | 3.2.4 |  |  |
| Use case view. | 3.3.1 |  | 3.3.1 |  |  |  |