SOFTWARE REQUIREMENTS SPECIFICATIONS

International Institute of Information Technology, Bangalore

Reactive Data System

Event Driven Architecture

Project Guide: Prof. Chandrasekhar Ramanathan

Team Number : 50

Team Details:

Names	Roll Number
Bisen Vikrantsingh Mohansingh	MT2012036
Kodamasimham Pridhvi	MT2012066
Vaibhav Singh Rajput	MT2012155

Table of Contents

Revision History

Date	Version	Description	Author
17/02/2013	1.0	Initiation of Project	Team 50

Introduction

Purpose

The aim of this project is to develop Reactive data System based on Event Driven Architecture which promotes production, detection, consumption and reaction to the events. EDA will act as a framework, on which event based applications will run.

Scope

The EDA Architecture will help the application developer to develop event based applications.

- Architecture will provide API to developer's application.
- EDA will support Database and Time based event.
- EDA will tell the application about the event occurred, so that corresponding action can be performed.
- EDA can be used to implement any event based application as its generic.
- EDA will support more than one event with multiple actions.

Document Conventions

- UPPERCASE LETTERS = INDICATES MOST NECESSARY REQUIRMENT
- Bold font style = Indicates actors/end users
- Italic font style = optional features
- EDA = Event Driven Architecture
- RDS = Reactive Data System
- Developer = Application Developer going to use EDA API
- For more refer Appendix B: Glossary

Intended Audience and Reading Suggestion

This document is designed for all those who are involved in the software development process directly or indirectly to provide information namely features and functionality of EDA.

- For Project manager: Looking at whole document will be beneficial but major stress should be on Non Functional Requirements.
- For Marketing staff: should refer Product feature and Non functional requirement of the artifact.

- For Developers: must follow Analysis model from System features with External interfaces
- For Tester: System features and external interfaces.
- For Documentation writers: System features.
- For Users: End user must read Product functions, User documentation, Assumptions and dependencies and Software quality attributes.

Overall Description

Product Perspectives

Developer will use EDA's API to register the event, and Listener will keep checking if that event occurred, and will inform the application so that it can take the corresponding action.

EDA will be build on top of trigger by extending it to overcome its disadvantages. The disadvantages of trigger are:

- Triggers do not support transactions.
- Triggers cannot call external function whenever any event occurs.

Product Functions

- EDA will have Listener, which will continuously check for event to occur.
- EDA will have two repositories Event Type repository and Event repository. Event-Type repository will be used to register events and Event repository will have the actual data related to the event.
- EDA Architecture is build over the trigger's which doesn't support transactions. So we are overcoming this drawback of triggers.
- Individual thread will be created for handling respective action of the event.

Operating Environment

- Technology to be used:
  - UML
  - J2EE
  - AJAX
  - Web 2.0

- Tools to be used:

- Eclipse IDE
- Umbrello for UML
- Apache Application Server
- Mysql Database Server
- Win 7 Operating System

Design and Implementation Constraints

- Do's
  - Simple user graphical user interface (GUI).
  - The files in which the information regarding securities and portfolios should be secured against malicious deformations.
  - Data should not become corrupted in case of system crash or power failure.
  - Support Re-Engineering and Component Based Development to save time and money.
- Don'ts
  - Use of platform dependent tools should be prohibited.
  - Development from scratch.

User Documentation

- Our web application needs high documentation both for user and developer/tester.
- Online help for all kinds of user will be available differently.
- As well as help document is provided in the form of .pdf user manual.
- It contains the topic from installation to maintenance for the work product.

Assumptions and Dependencies

- Database and web server should work properly and compatible with different system.
- Site's resolution should be suitable considering varying browser types.
- The administrator should be well experienced of handling computer system.

System Features

EDA Features

- Provides different points of view for application developer.
- In this project we are trying to build an application which is highly reactive to data modification in RDBMS. For this we need to develop a Event Driven Framework which will consist of mainly three components Event, Dispatcher and Action.
- User will define events and respective action. Dispatcher will run as a daemon process and check for events continuously if any event occurs then respective action is performed.
- It has many applications in scenario where system has to perform action asynchronously depending on incoming data or time based events. Some of example would be setting minimum threshold value of shares to purchase automatically.
- Our EDA will be generic framework which can be used by almost any event driven application. For demonstrating our RDS we've selected admission process as our domain.

EDA Architecture

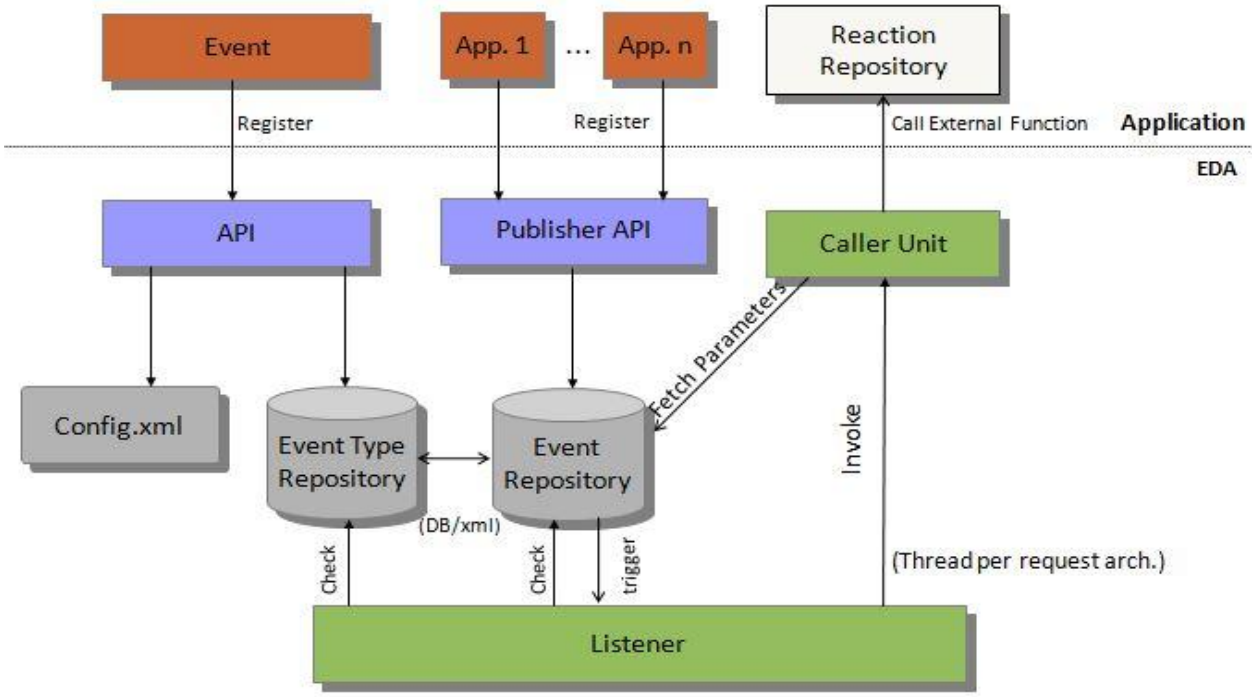


Fig. RDS over EDA Architecture

- Event
    - An event can be defined as "a significant change in state".
    - Here in this context Event can be taken as a user action or a time based action taking place.
    - For example, User clicks on submit button is an event and closing of registration page after closing date.
- API
    - It acts as a communication medium between Application and EDA
    - Application developer will make use of api to register event and respective action.
    - Also provide database configuration, etc information through API
- Publisher API
    - Application uses this API to interact with event repository where actual data manipulation takes place.
- Event Repository
    - Store event and respective reaction details.
- Event Type Repository
    - This is the place where actual data driven event happens.
    - Application uses this place to store its data.
- Listener
    - Listener is the channel for calling the respective reaction for the occurrences of the respective event as specified by the developer.
    - Here a particular event is mapped to its respective reaction and a thread is created to carry on the reaction.
    - Whenever the reaction is completed the thread is automatically deleted.
- Caller Unit
    - Based on thread per request architecture (i.e. new thread is created for each request call from listener to execute function at app layer in reaction repository)
- Reaction Repository
    - It consists of all multiple packages of classes and functions which are registered as a reaction for some specific events.

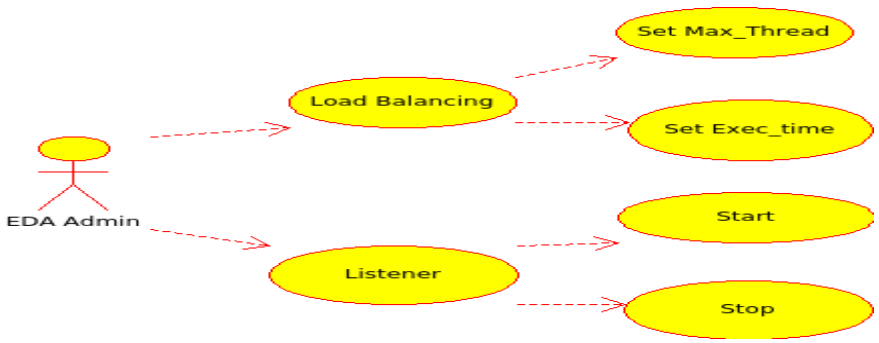
Application Features

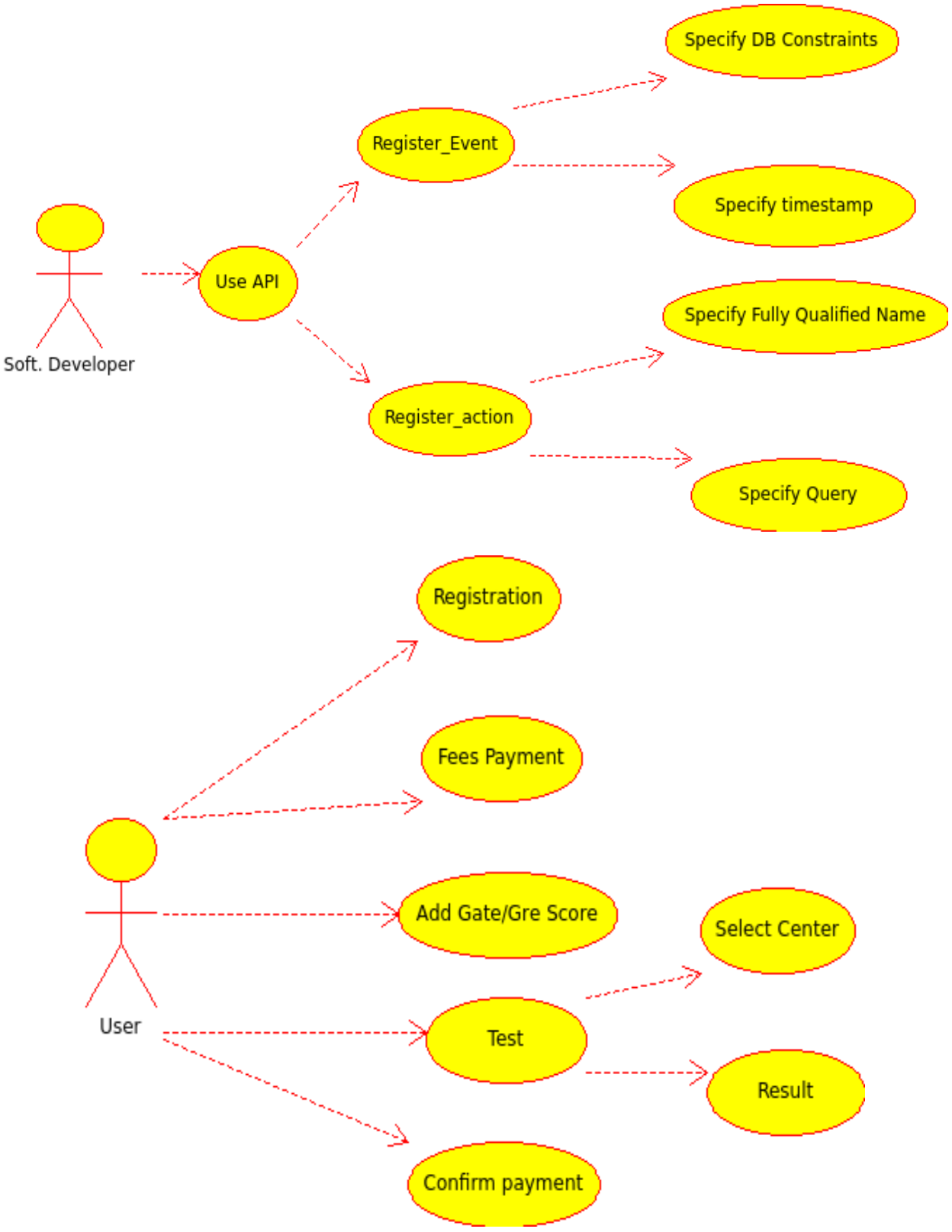
- We have chosen Admission Process Automation as application to demonstrate working of our EDA framework.
- We have identified following event in this process which can be registered with respective reaction and

- Registration (Event)
  - Auto Mail (Data driven Action)
  - Auto Registration Close – (Time based Action)
- Fees Payment (Event)
  - Paid / Pending
  - Auto Mail
  - Make Eligible for
- Set Eligible for Test
- Add GATE / GRE Score
- Apply of waiver
- Cutoff
- Mail, Status change(selected or rejected)
- Entrance Test
- Center Allocation
- Result
- Interview Call

Analysis Models

Use Case Diagram




Specify DB Constraints
Register_Event
Specify timestamp
Soft. Developer
Use API
Specify Fully Qualified Name
Register_action
Specify Query
Registration
Fees Payment
Add Gate/Gre Score
Select Center
User
Test
Result
Confirm payment

Database Schema Diagram

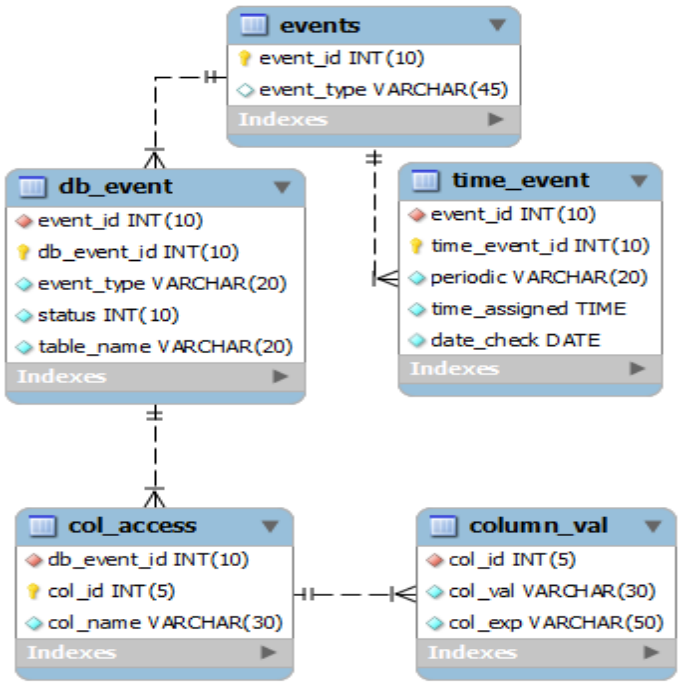


Fig. Database Schema for Event Type Repository Part - I

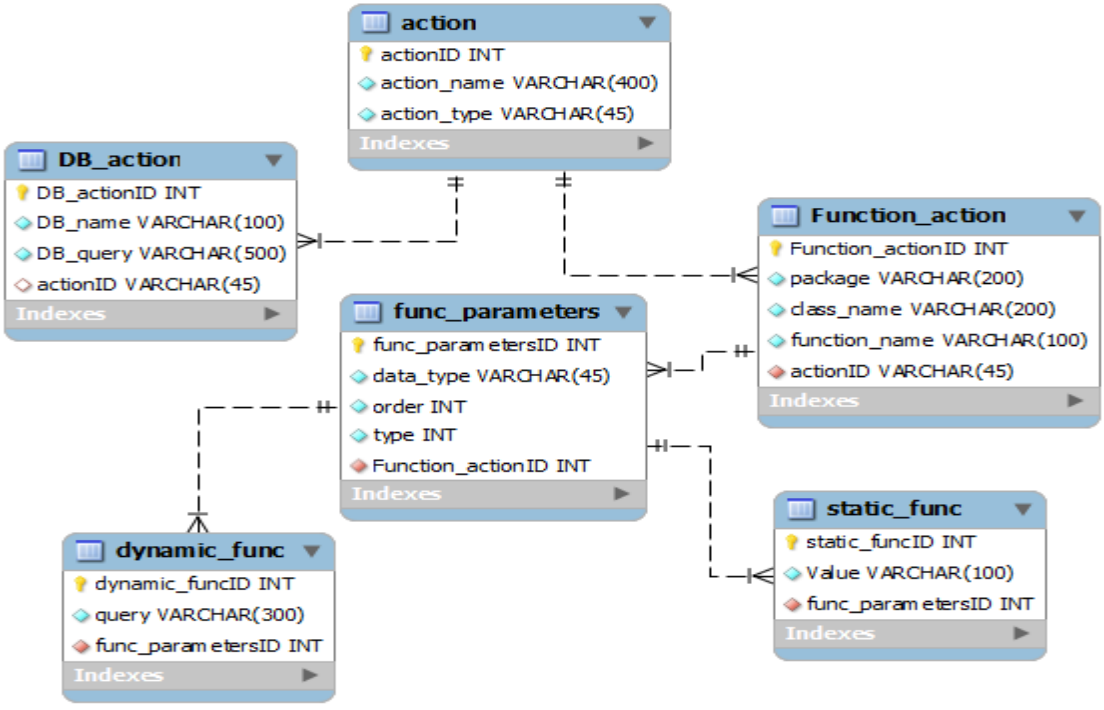
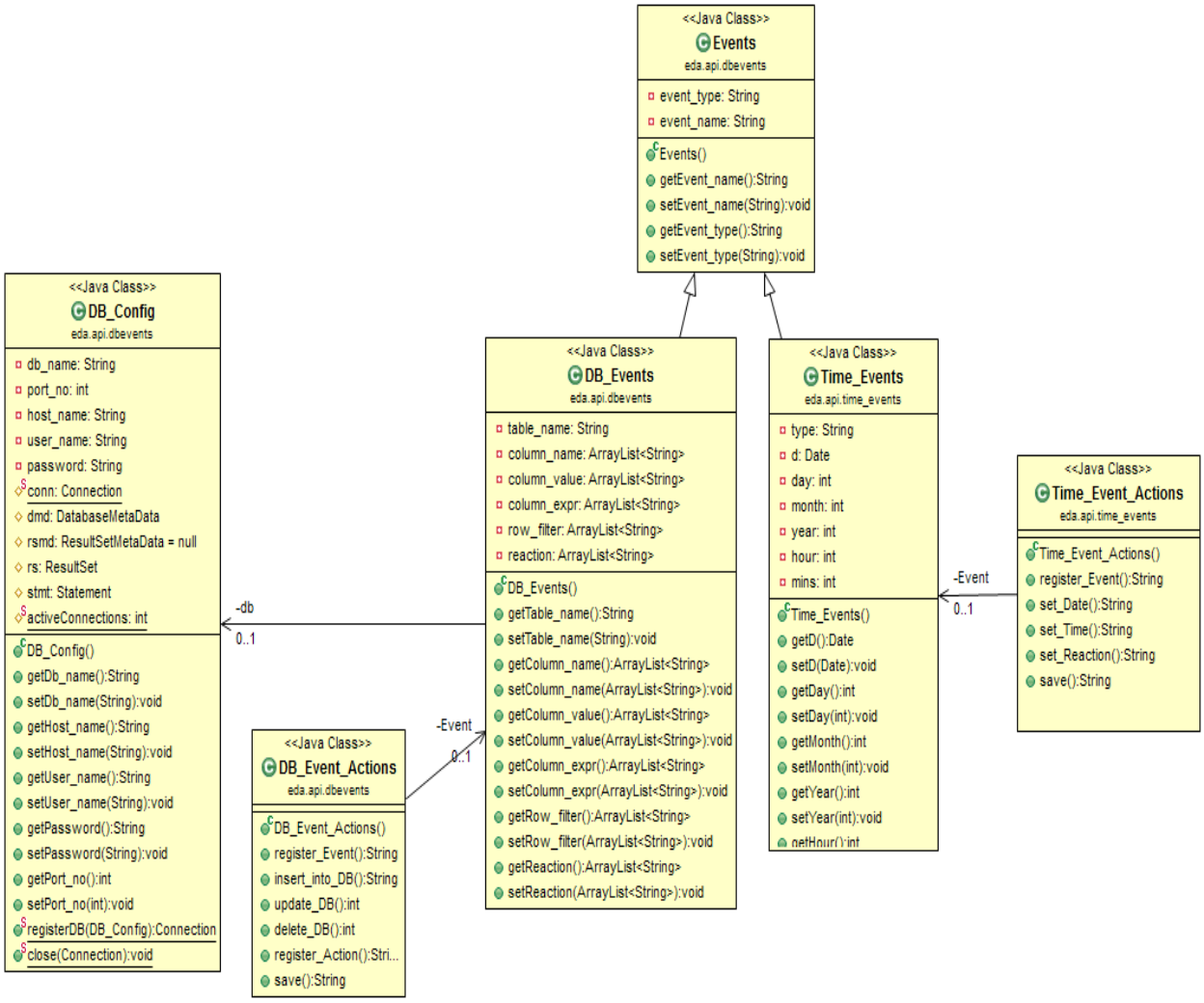
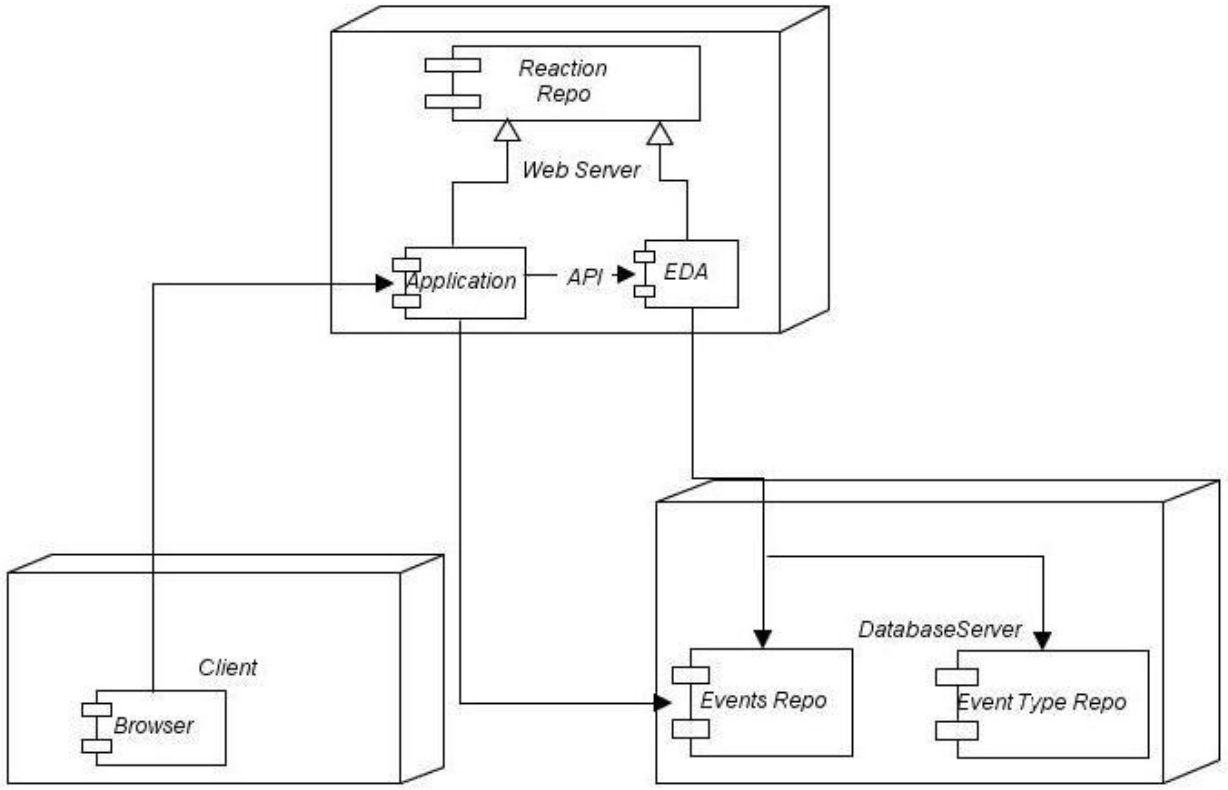


Fig. Database Schema for Event Type Repository Part - II

API Module Class Diagram



Deployment Diagram



External Interface Requirements

User Interfaces

- Simple and attractive graphical user interface is needed.
- Easy to Use.
- Light weight.
- Appendix A shows the intended user screen.

Software Interfaces

- To develop the application kindly makes use of multiple open source products.
- MYSQL should be used to store and manage the database effectively.

Communications Interfaces

- Provide API's for registration of events and reactions to application developer.
- Try to build driver package to make app database independent.

Non Functional Requirements

Performance Requirements

- System should run on any operating system.
- The database server may change in future so precaution to be taken.
- Provision for load balancing.
- Flexible and scalable in future.

Security Requirements

- Monitor threads for running functions of repository reaction unit.
- Provide encryption of data wherever possible.

Software Quality Attributes

- Available for 24X7
- Error/Bug Free
- Flexible for modification
- High documentation for Easy to maintenance.
- Portable on any platform.

Appendix A

Sample User Interface

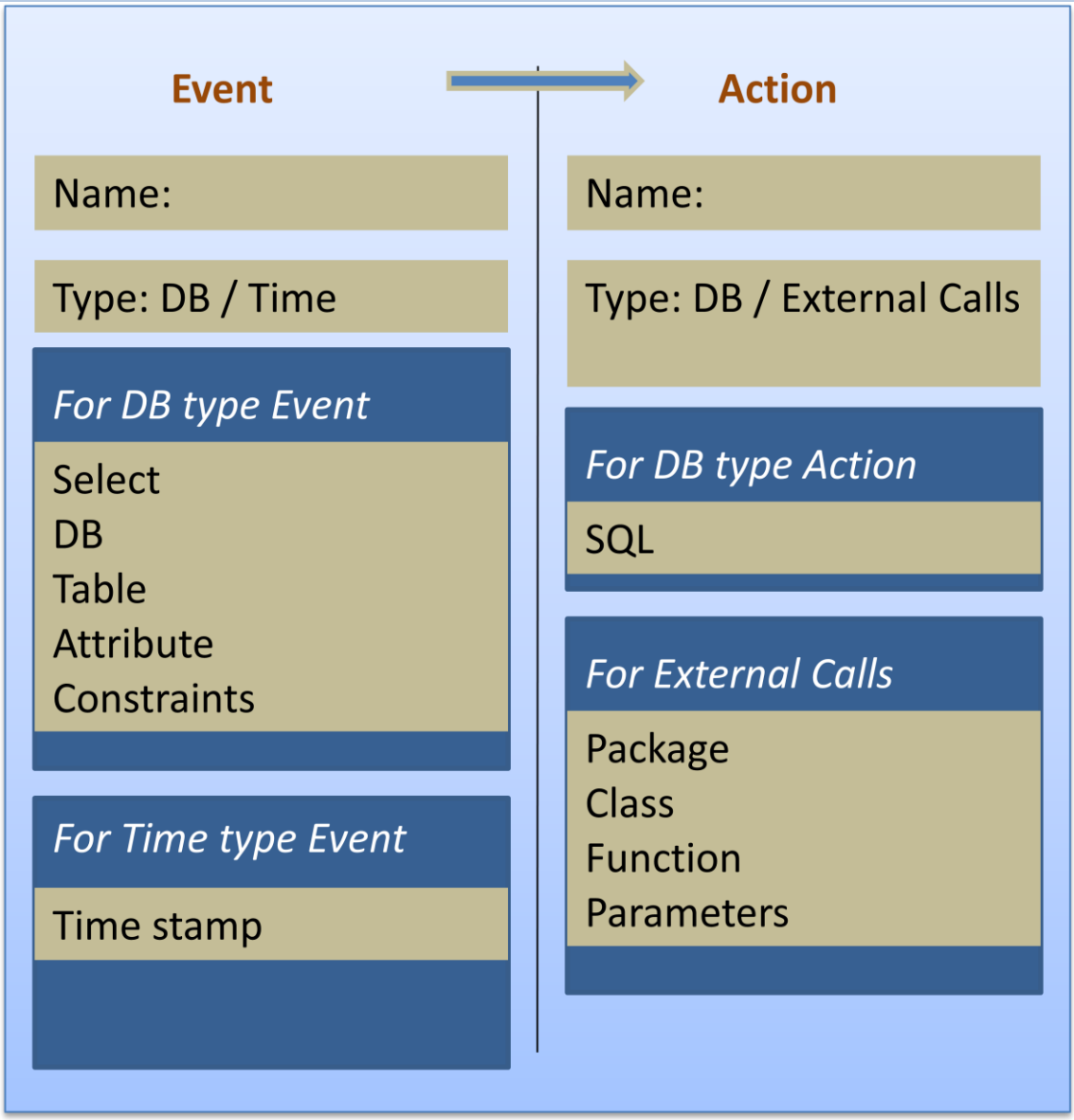


Fig. Sample Event Registration Form

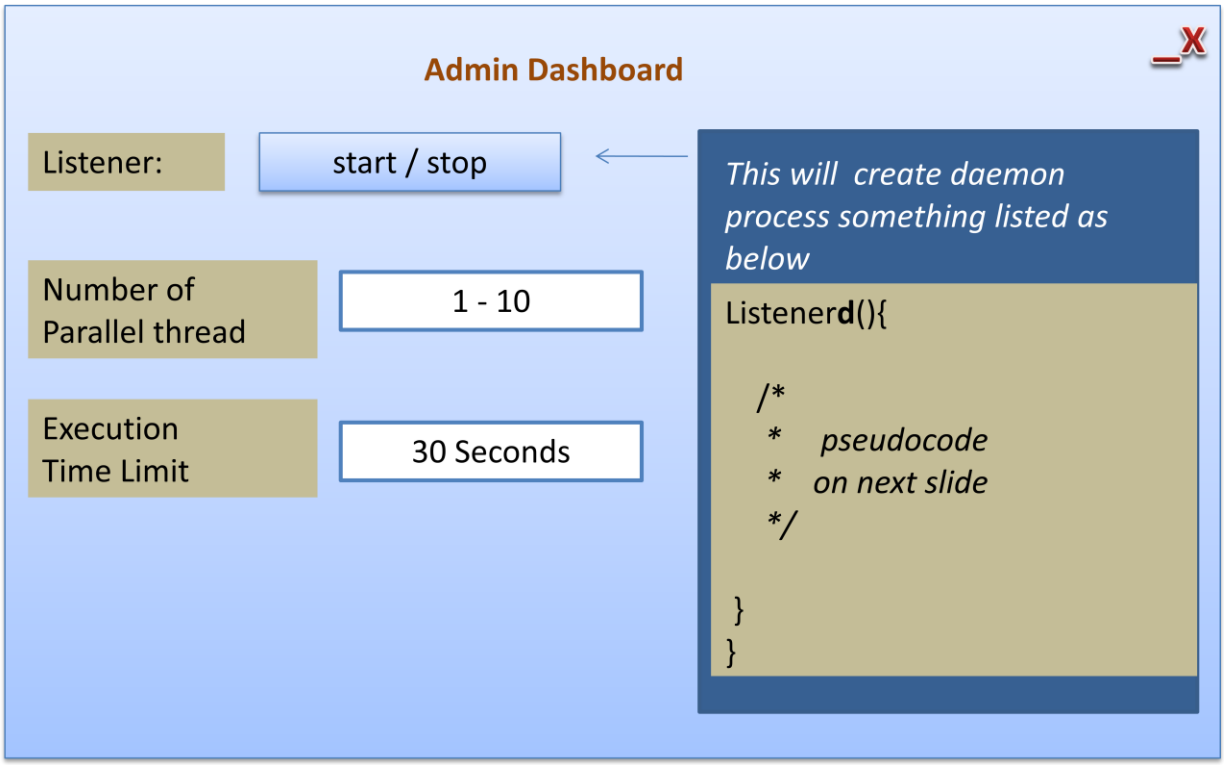


Fig. Sample Admin Dashboard

Appendix B

Glossary

- *Event* - An event can be defined as "a significant change in state".
- *Action* – Action can be an SQL query or a fully qualified class name assigned to a particular occurrence of an event.
- *Reaction* – Reaction is the result obtained by executing/performing the respective action when an event occurs.
- *Admin* – Administrator is the user who can control the complete EDA.
- *EDA* – Event Driven Architecture
- *RDS* – Reactive Data System
- *App Developer* – Person who uses our software for handling events from his application.

- *API* - An **Application Programming Interface** (**API**) is a protocol intended to be used as an interface by software developer to communicate with Event-Type repository and Event repository.