Software Design Description for FRC Scout

Version 1.0

March 5, 2013

Prepared by:

Kristian Calhoun, Hannah Pinkos, Keith Horrocks, Ryan Hersh, Jirakit Songprasit

Drexel University

Revision History

Name	Date	Reason for Changes	Version
Kristian Calhoun, Hannah Pinkos, Keith Horrocks, Ryan Hersh, Jirakit Songprasit	March 5, 2013	Initial Version	1.0

Contents

1. Introduction

1.1 Background

This document specifies the entire software architecture for the FRC Scout system. The design decisions outlined below were made in compliance with the software constraints and functionality requirements outlined in the FRC Scout Software Requirements Specification document.

FRC Scout allows users, who are members of a single FIRST Robotics Competition (FRC) team, to input data about the performance of each robot competing at a FRC event into a graphical user interface. The program stores the information and computes simple statistics. The system allows users to view the data in a graphical format and search for and sort the information based on different user-selected criteria. Users must log into the system and, depending on their user role, have access to view or input different information in the system.

1.2 Scope

This document describes the software architecture for the initial release of FRC Scout, version 1.0. The implementation of FRC Scout will adhere to the models and design decisions outlined in this report. The intended audience of this document includes the designers, developers and testers of the FRC Scout system.

1.3 Glossary

Administrator An administrator is a user with the “administrator” account privilege. In addition to the team member and scout user permissions, administrators can modify or delete any data in the system, add new events to the system, and manage user accounts.

Event An event is a single regional competition or championship division. Each event is defined by its date, name, and list of attending teams.

Java A class-based, object-oriented, platform independent programming language

JavaBean A Java class that has a zero argument constructor and methods that allow properties and methods to be accessed and controlled by another application (often JSP pages).

JavaScript A scripting programming language primarily used in the development of dynamic web content.

JavaServer Pages (JSP) A technology that allows for the dynamic creation of HTML web pages. They require a compatible web server with a servlet container to run.

Java Database Connectivity (JDBC) An API that defines database access from a Java program.

Hypertext Markup Language (HTML) HTML is the primary markup language used to display web pages in web browsers..

Internet The global network of interconnected computers that use the same set of communication protocols.

Match In the context of FRC Scout, a match refers to a single qualification match played at an event. Each match has a match number and matches are played sequentially at an event. A match can also be identified by the six teams playing in it.

Match Record A match record is a unique collection of data relevant to the game for a particular team in a particular match at a particular event.

Model-View-Controller (MVC) An architecture that allows for the separation of data representation, data visualization, and data manipulation.

Scout A scout is a user with the “scout” account privilege. In addition to having team member permissions, a scout can also input new data into the system or modify previously entered data.

Servlet A Java class that extends the functionality of a server.

Team A team refers to a group of people who field a single robot at an event. All teams are assigned an official, permanent team number upon registering for the FIRST Robotics Competition. Each team also has a team nickname by which it can be recognized.

Ultimate Ascent Ultimate Ascent is the name of the 2013 FIRST Robotics Competition (FRC) game in which robots score flying discs into goals of varying heights and climb a pyramid structure to earn points.

User A user is a person who owns an account in the FRC Scout system.

Web server A computer application that delivers web content that can be access through a network (usually the Internet).

1.4 Context Diagram

The context diagram in Figure 1 shows a simplified, high level overview of the main components of FRC Scout.

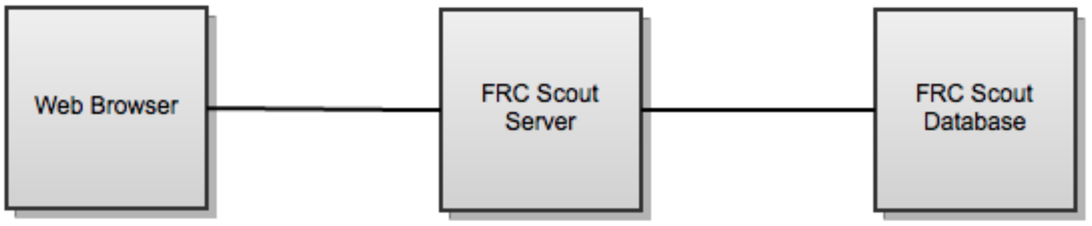


Figure 1. *Context Diagram. A user's web browser communicates with the FRC Scout Server, which handles user operations and retrieves and modifies data stored in the FRC Scout database.*

2. Architecture

2.1 Overview

As a dynamic web application, FRC Scout makes use of a model-view-controller (MVC) architecture. An MVC architecture allows for the separation of data representation, data visualization, and data manipulation. Therefore different components in the system each maintain a single responsibility. FRC Scout benefits from an MVC architecture because one of the primary uses of the system is to allow users to visualize and manipulate the same data in multiple views. The MVC architecture assists the development of this functionality because the model and the view are decoupled.

2.2 Technologies Used

FRC Scout will use a MySQL database to store information about events, team, and users in the system. An Apache Tomcat server will serve users JavaServer Pages (JSP) that display information queried from the database. In addition to JSP, the served pages will make use of a third-party JavaScript library, ExtJS, to assist in creating the view of the data on the delivered pages.

In terms of the MVC architecture, the JavaBeans classes act as the models, some JSP web pages (with the help of ExtJS scripts) act as the views, and other JSP pages communicating with the server act as the controllers.

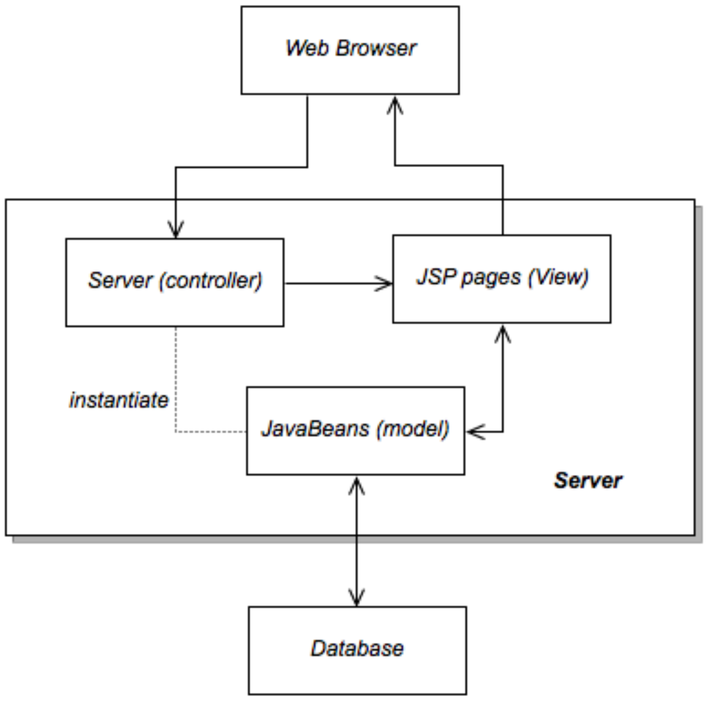


Figure 2.1 *Model View Controller Architecture applied to FRC Scout*

2.3 System Architecture

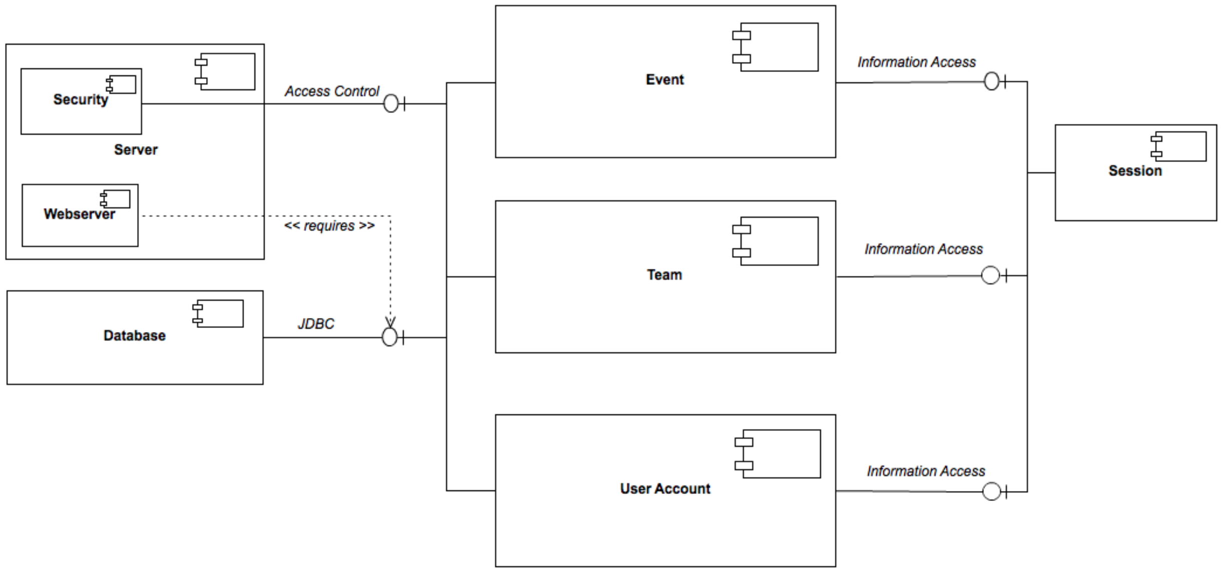


Figure 2.2. *Component Diagram*

2.4 Server Components

Requirements Satisfied: 0120, 0200, 0230 - 0275, 0290, 0370, 0400, 0420, 0480, 0510, 0530, 0810, 0850, 0870,

The server handles user authentication and serving the JSP pages. Additionally, the Access Control interface ensures that only pages visible to a user with the proper user role are delivered. For example, a user with the Team Member or Scout permissions cannot view the administrator management pages and will receive an error if they attempt to access them. The server requires a Java Database Connection to the system's database.

2.5 User Account Components

Requirements Satisfied: 0000-0110, 0130-0190, 0210-0220

The user account components manage the movement of user account data between the web application user interface and the database. It contains related JavaBean classes that store and distribute user account information retrieved from the database. These also contain methods that insert and modify data in the database (Figure 4). The classes also include the validation of user input before entering it into the database. This component accesses the database by interfacing with JDBC.

2.6 Team Components

Requirements Satisfied: 0410, 0430-0470, 0480-0500

The team component manages the movement of team data between the web application user interface and the database. It contains related JavaBean classes that store and distribute team information retrieved from the database. It also contains methods that insert and modify data in the database (Figure 4). This includes the validation of user input before entering it into the database. This component accesses the database by interfacing with JDBC.

2.7 Event Components

Requirements Satisfied: 0280, 0300-0360, 0380-0390, 0520, 0540-0800, 0820-0840, 0860-0980

The event component manages the movement of event data, including data related to matches at the event, between the web application user interface and the database.   It contains related JavaBean classes that store and distribute event information retrieved from the database. It also contains methods that insert and modify data in the database (Figure 4). This includes the validation of user input before entering it into the database. This component accesses the database by interfacing with JDBC.

2.8 Session Components

Requirements Satisfied: 0270 - 0275

The session component contains information about the user that is currently logged into the system. By interfacing with the event, team and user account components, logged in users are able to access and control information stored in the FRC Scout database.

3. Data Model

3.1 Class Diagram

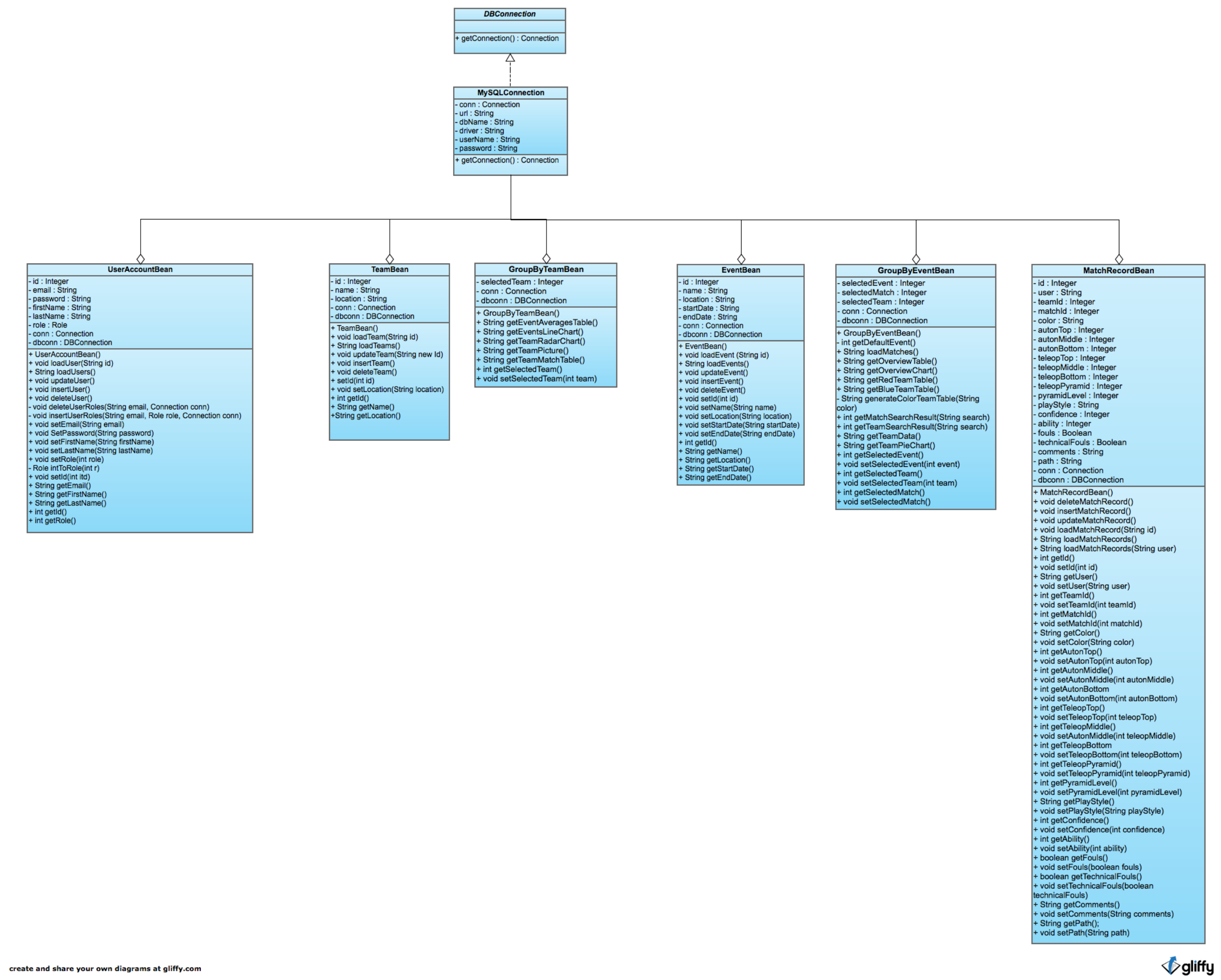


Figure 3. *Class Diagram*

3.2 Class Descriptions

3.2.1 DBconnection

DBConnecion is an interface that provides the prototype for the public function getConnection(), which returns a Connection object.

3.2.2 MySQLConnection

MySQLConnection implements the DBConnection interface, and is primarily used to create and return a connection to the MySQL database.

3.2.2.1 Attributes

Name	**Type**	**Description**
conn	Connection	A connection to the MySQL database.
url	String	The URL of the database to connect to.
dbName	String	The name of the database to connect to.
driver	String	The package name of the database driver (e.g. com.mysql.jdbc.Driver).
userName	String	The username to use to connect to the database.
password	String	The password to use to connect to the database.

3.2.2.2 Methods

Name:	Connection getConnection()
Input:	void
Output:	Returns a Connection to the FRC Scout Server Database
Description:	The purpose of this method is to have a single pathway for all connections to FRC scout Server Database.

3.2.3 UserAccountBean

The primary purpose of UserAccountBean is to insert, update, delete and query from the user related tables of the database.

3.2.3.1 Attributes

Name	**Type**	**Description**
id	Integer	A unique user id.
email	String	An email address of the form username@domain.ext to be used to login to the FRC Scout system.

password	String	An alphanumeric password to be used for logging into the FRC Scout system.
firstName	String	The user’s first name.
lastName	String	The user’s last name.
role	Role	The user’s permission role (either Role.administrator, Role.scout, or Role.team_member).
conn	Connection	A database connection
dbconn	DBConnection	Specifies which database is to be connected to

3.2.3.2 Methods

Name:	UserAccountBean()
Input:	void
Output:	void
Description:	Constructs a new UserAccountBean object with default attributes.

Name:	void loadUser(String id)
Input:	A string containing the id of a user in the database.
Output:	void
Description:	Queries the database for a user with the given id and loads the data into the attributes.

Name:	String loadUsers()
Input:	void
Output:	a String, formatted as a JSON Array
Description:	Queries the database for all active users and returns a String containing the data.

Name:	void updateUser()
Input:	void

Output:	void
Description:	Updates the entry in the database based on the values currently in the attributes

Name:	void insertUser()
Input:	void
Output:	void
Description:	Inserts a new user with this object's attributes' values into the database.

Name:	void deleteUser()
Input:	void
Output:	void
Description:	Deactivates the account for the user id in the id attribute.

Name:	void deleteUserRoles(String email, Connection conn)
Input:	● A String representing a user's email address. ● A Connection to the FRC scout Server Database.
Output:	void
Description:	Removes all permissions for the given user.

Name:	void insertUserRoles(String email, Role role, Connection conn)
Input:	● A String representing a user's email address. ● The Role to grant the user. ● A Connection to the FRC scout Server Database.
Output:	void
Description:	Grants the given user the permissions associated with the provided user role.

Name:	void setEmail(String email)
Input:	A user's email address.

Output:	void
Description:	Sets the email attribute after validation of the String parameter

Name:	void setPassword(String password)
Input:	A String representing a new password.
Output:	void
Description:	Sets the password attribute after validation of the String parameter

Name:	void setFirstName(String firstName)
Input:	A String representing the user's first name.
Output:	void
Description:	Sets the firstName attribute after validation of the String parameter

Name:	void setLastName(String LastName)
Input:	A String representing a user's last name.
Output:	void
Description:	Sets the lastName attribute after validation of the String parameter.

Name:	void setRole(int role)
Input:	An integer corresponding to a given user role (1: team member, 2: scout, 3: administrator).
Output:	void
Description:	Sets role attribute after validation of the int parameter

Name:	Role intToRole(int r)
Input:	An integer corresponding to a user role.
Output:	The Role corresponding to the provided integer.
Description:	Converts an integer to its corresponding user role.

Name:	void setId(int id)
Input:	An integer representing the user’s new ID.
Output:	void
Description:	Sets the id attribute

Name:	String getEmail()
Input:	void
Output:	A string representing the email address of a user.
Description:	Returns the email address of the currently active user.

Name:	String getFirstName()
Input:	void
Output:	A string representing the first name of a user.
Description:	Returns the first name of the currently active user.

Name:	String getLastName()
Input:	void
Output:	A string that representing the last name of user.
Description:	Returns the last name of the currently active user.

Name:	int getId()
Input:	void
Output:	An integer representing the id of user.
Description:	Returns the ID of the currently active user.

Name:	int getRole()
Input:	void
Output:	An integer representing the role of a user.
Description:	Returns the numerical representation of the currently active user’s user

	role.

3.2.4 TeamBean

The primary purpose of TeamBean is to insert, update, delete and query from the team related tables of the database.

3.2.4.1 Attributes

Name	**Type**	**Description**
Id	Integer	A team’s unique team number.
Name	String	A team’s nickname.
location	String	The team’s geographic location.
conn	Connection	A database connection
dbconn	DBconnection	Specifies which database is to be connected to

3.2.4.2 Methods

Name:	TeamBean()
Input:	void
Output:	void
Description:	Constructs a new TeamBean object with default attributes.

Name:	void loadTeam(String id)
Input:	A team’s unique team number.
Output:	void
Description:	Queries the database for the given team and sets the attributes of the TeamBean object with the resulting data

Name:	String loadTeams()

Input:	void
Output:	A String, formatted as a JSON array
Description:	Queries the database for all teams and returns a String containing the data

Name:	void updateTeam(String newId)
Input:	The team number of the team to update.
Output:	void
Description:	Updates the information for a given team in the database with the current values of this object's attributes.

Name:	void insertTeam()
Input:	void
Output:	void
Description:	Inserts a team with this object's attributes into the FRC Scout database.

Name:	void deleteTeam()
Input:	void
Output:	void
Description:	Deletes the team with the id stored in the id attribute from the database.

Name:	void setId(int id)
Input:	A team number.
Output:	void
Description:	Sets the team number for the currently loaded team.

Name:	void setLocation(String location)
Input:	A String containing the location of a team.
Output:	void
Description:	Sets the location of the currently loaded team.

Name:	int getId()
Input:	void
Output:	An integer that representing the id of a team.
Description:	Returns the team number for the currently loaded team.

Name:	String getName()
Input:	void
Output:	A string representing the name of a team.
Description:	Returns the name of the currently loaded team.

Name:	String getLocation()
Input:	void
Output:	A String representing the location of a team.
Description:	Returns the location of the currently active team.

3.2.5 EventBean

The primary purpose of EventBean is to insert, update, delete and query from the Event related tables of the database.

3.2.5.1 Attributes

Name	**Type**	**Description**
Id	Integer	The unique ID of an event.
Name	String	The name of the event.
location	String	The geographic location of the event.
startDate	String	A String representing the start date of the event.

endDate	String	A String representing the end date of an event.
conn	Connection	A database connection
dbconn	DBconnection	Specifies which database is to be connected to

3.2.5.2 Methods

Name:	EventBean()
Input:	void
Output:	void
Description:	Constructs a new EventBean object with default attributes.

Name:	void loadEvent(String id)
Input:	The unique ID of the event to be loaded.
Output:	void
Description:	Queries the database for the given event id, then sets the attributes of this EventBean object with its details.

Name:	String loadEvents()
Input:	void
Output:	String, formatted as a JSON Array
Description:	Queries the database for all events and returns this data

Name:	void updateEvent()
Input:	void
Output:	void
Description:	Updates the information for a given event in the database with the current values of this object’s attributes.

Name	void insertEvent()
Input:	void
Output:	void
Description:	Inserts an event with this object’s attributes into the FRC Scout database.

Name	void deleteEvent()
Input:	void
Output:	void
Description:	Deletes the event with this object’s id from the database.

Name:	void setId(int id)
Input:	A unique numerical ID for the event.
Output:	void
Description:	Sets the ID attribute for the currently loaded Event.

Name:	void setName(String name)
Input:	A String representing the name of an event.
Output:	void
Description:	Sets the name attribute for the currently loaded event.

Name:	void setLocation(String location)
Input:	A String representing the location of an event.
Output:	Void
Description:	Sets the location attribute for the currently loaded event.

Name:	void setStartDate(String startDate)
Input:	A String representing the start date of the event
Output:	void

Description:	Sets the startDate attribute for the currently loaded event.

Name:	void setEndDate(String endDate)
Input:	A String representing the end date of the event
Output:	void
Description:	Sets the endDate attribute for the currently loaded event.

Name:	int getId()
Input:	void
Output:	An integer representing the id of event.
Description:	Returns an integer representing the ID of the currently loaded event.

Name:	String getName()
Input:	void
Output:	A string representing the name of an event.
Description:	Returns the name of the currently loaded event.

Name:	String getLocation()
Input:	void
Output:	A string representing the location of an event.
Description:	Returns the geographic location of the currently loaded event.

Name:	String getStartDate()
Input:	void
Output:	A string representing the start date of an event.
Description:	Returns the start date for the currently loaded event.

Name:	String getEndDate()

Input:	void
Output:	A string representing the end date of an event.
Description:	Returns the end date for the currently loaded event.

3.2.6 MatchRecordBean

The primary purpose of MatchRecordBean is to insert, update, delete and query from the match and match record related tables of the database.

3.2.6.1 Attributes

Name	Type	Description
id	Integer	The match record’s unique ID.
user	String	The email address of the user who created the match record.
teamId	Integer	The team number of the team the match record contains data for.
matchId	Integer	The match number of the match at the event where it took place.
eventId	Integer	The ID of the event where the match took place.
color	String	The alliance color (red or blue) for the team during this match.
autonTop	Integer	The number of discs scored by the team in the top goal during autonomous.
autonMiddle	Integer	The number of discs scored by the team in the middle goal during autonomous.
autonBottom	Integer	The number of discs scored by the team in the bottom goal during autonomous.
teleopTop	Integer	The number of discs scored by the team in the top goal during the teleoperated period.
teleopMiddle	Integer	The number of discs scored by the team in the middle goal during the teleoperated period.

teleopBottom	Integer	The number of discs scored by the team in the bottom goal during the teleoperated period.
teleopPyramid	Integer	The number of discs scored by the team in the pyramid goal during the teleoperated period.
pyramidLevel	Integer	The level of the pyramid a robot climbed to (0, 1, 2, or 3).
playStyle	String	A String representing the team's primary style of play during the match: either "offense" or "defense".
confidence	Integer	The scout's personal rating of the team based on their performance during the match (0-10).
ability	Integer	A number, 0-5, rating the team's ability to pick up discs from the ground.
fouls	Boolean	True if the robot committed any fouls during the match, false otherwise.
technicalFouls	Boolean	True if the robot committed any technical fouls during the match, false otherwise.
comments	String	Any notes the scout chooses to enter about a team based on their performance during a match.
path	String	A description of the path the team's robot drove during the autonomous period.
conn	Connection	A database connection
dbconn	DBConnection	Specifies which database is to be connected to.

3.2.6.2 Methods

Name:	MatchRecordBean()
Input:	void
Output:	void
Description:	Constructs a new MatchRecordBean with default attribute values.

Name:	void deleteMatchRecord()
Input:	void

Output:	void
Description:	Deletes a match record with the object’s id from the database.

Name:	void insertMatchRecord()
Input:	void
Output:	void
Description:	Inserts a match record to the database with the object’s attributes.

Name:	void updateMatchRecord()
Input:	void
Output:	void
Description:	Updates an existing match record database entry corresponding to the object’s id to match the object’s attributes.

Name:	void loadMatchRecord(String id)
Input:	A String representing the unique ID for a given match record.
Output:	void
Description:	Queries the database for an entry with the given id, then fills the attributes of the object with the data.

Name:	String loadMatchRecords()
Input:	void
Output:	A String formatted as a JSON Array containing all match records for all events.
Description:	Queries the database for all match records for all events.

Name:	String loadMatchRecords(String user)
Input:	A string representing the email address of the user whose match records are to be retrieved
Output:	A String formatted as a JSON Array containing all match records for all events entered by the given user.

Description:	Queries the database for all match records for all events.

Name:	int getId()
Input:	void
Output:	The unique integer ID for the match record.
Description:	Returns the ID of the match record.

Name:	void setId(int id)
Input:	An integer ID.
Output:	void
Description:	Sets the ID of the match record.

Name:	String getUser()
Input:	void
Output:	A String representing the creating user’s email address
Description:	Returns the creating user’s email address for the match record.

Name:	void setUser(String user)
Input:	A String representing the creating user’s email address
Output:	void
Description:	Sets the creating user’s email address of the match record.

Name:	int getTeamId()
Input:	void
Output:	The team number for the currently selected match record
Description:	Retrieves the team number for the currently selected match record.

Name:	void setTeamId(int teamId)

Input:	The new team number, an integer, for the active match record.
Output:	void
Description:	Sets the team number the match record is about.

Name:	int getMatchId()
Input:	void
Output:	A match ID
Description:	Returns the ID of the match the match record is about.

Name:	void setMatchId(int matchId)
Input:	A match ID
Output:	void
Description:	Sets the ID of the match the match record is about.

Name:	String getColor()
Input:	void
Output:	A String representation of a color
Description:	Returns the assigned alliance color of the team the match record is about.

Name:	void setColor(String color)
Input:	A string representation of a color
Output:	void
Description:	Sets the assigned alliance color of the team the match record is about.

Name:	int getAutonTop()
Input:	void
Output:	An integer count

Description:	Returns the number of shots made in the top goal during the autonomous period by the team the match record is about.

Name:	void setAutonTop(int autonTop)
Input:	An integer count
Output:	void
Description:	Sets the number of shots made in the top goal during the autonomous period by the team the match record is about.

Name:	int getAutonMiddle()
Input:	void
Output:	An integer count
Description:	Returns the number of shots made in the middle goal during the autonomous period by the team the match record is about.

Name:	void setAutonMiddle(int autonMiddle)
Input:	An integer count
Output:	void
Description:	Sets the number of shots made in the middle goal during the autonomous period by the team the match record is about.

Name:	int getAutonBottom()
Input:	void
Output:	An integer count
Description:	Returns the number of shots made in the bottom goal during the autonomous period by the team the match record is about.

Name:	void setAutonBottom(int autonBottom)
Input:	An integer count
Output:	void

Description:	Sets the number of shots made in the bottom goal during the autonomous period by the team the match record is about.

Name:	int getTeleopTop()
Input:	void
Output:	An integer count
Description:	Returns the number of shots made in the top goal during the teleoperated period by the team the match record is about.

Name:	void setTeleopTop(int teleopTop)
Input:	An integer count
Output:	void
Description:	Sets the number of shots made in the top goal during the teleoperated period by the team the match record is about.

Name:	int getTeleopMiddle()
Input:	void
Output:	An integer count
Description:	Returns the number of shots made in the middle goal during the teleoperated period by the team the match record is about.

Name:	void setTeleopMiddle(int teleopMiddle)
Input:	An integer count
Output:	void
Description:	Sets the number of shots made in the middle goal during the teleoperated period by the team the match record is about.

Name:	int getTeleopBottom()
Input:	void
Output:	An integer count

Description:	Returns the number of shots made in the bottom goal during the teleoperated period by the team the match record is about.

Name:	void setTeleopBottom(int teleopBottom)
Input:	An integer count
Output:	void
Description:	Sets the number of shots made in the bottom goal during the teleoperated period by the team the match record is about.

Name:	int getTeleopPyramid()
Input:	void
Output:	An integer count
Description:	Returns the number of shots made in the pyramid goal during the teleoperated period by the team the match record is about.

Name:	void setTeleopPyramid(int teleopPyramid)
Input:	An integer count
Output:	void
Description:	Sets the number of shots made in the pyramid goal during the teleoperated period by the team the match record is about.

Name:	int getPyramidLevel()
Input:	An integer pyramid level
Output:	void
Description:	Returns the level that the team’s robot climbed to.

Name:	void setPyramidLevel(int pyramidLevel)
Input:	void
Output:	An integer pyramid level
Description:	Sets the level that the team’s robot climbed to.

Name:	String getPlayStyle()
Input:	void
Output:	String description
Description:	Returns a String describing the play style the team had during the match.

Name:	void setPlayStyle(String playStyle)
Input:	String description
Output:	void
Description:	Sets the String description of the play style the team had during the match.

Name:	int getConfidence()
Input:	void
Output:	Integer rating
Description:	Returns an integer rating the team's performance during the match.

Name:	void setConfidence(int confidence)
Input:	Integer rating
Output:	void
Description:	Sets the rating of the team's performance during the match.

Name:	int getAbility()
Input:	void
Output:	Integer rating
Description:	Returns an integer rating the team's ability to pick up disks during the match.

Name:	void setAbility(int ability)
Input:	Integer rating
Output:	void
Description:	Sets the rating of the team’s ability to pick up disks during the match.

Name:	boolean getFouls()
Input:	void
Output:	True or False
Description:	Returns whether the team committed any fouls during the match.

Name:	void setFouls(boolean fouls)
Input:	True or False
Output:	void
Description:	Sets whether the team committed any fouls during the match.

Name:	boolean getTechnicalFouls()
Input:	void
Output:	True or False
Description:	Returns whether the team committed any technical fouls during the match.

Name:	void setTechnicalFouls(boolean technicalFouls)
Input:	True or False
Output:	void
Description:	Sets whether the team committed any fouls during the match.

Name:	String getComments()
Input:	void

Output:	Comment String
Description:	Returns the comments about the team’s performance during the match.

Name:	void setComments(String comments)
Input:	Comment String
Output:	void
Description:	Sets the comments about the team’s performance during the match.

Name:	String getPath()
Input:	void
Output:	String description
Description:	Returns a description of the path the robot took during the autonomous period of the match.

Name:	void setPath(String path)
Input:	String description
Output:	void
Description:	Sets the description of the path the robot took during the autonomous period of the match.

3.2.7 GroupByEventBean

The primary purpose of GroupByEventBean is to query the database for data related to a selected event.

3.2.7.1 Attributes

Name	**Type**	**Description**
selectedEvent	Integer	The unique ID of the selected

		event.
selectedMatch	Integer	The unique ID of the selected match.
selectedTeam	Integer	The unique ID of the selected team.
conn	Connection	A Database Connection
dbconn	DBConnection	Specifies which database is to be connected to

3.2.7.2 Methods

Name:	GroupByEventBean()
Input:	void
Output:	void
Description:	Constructs a new GroupByEventBean object with default attributes.

Name:	int getDefaultEvent()
Input:	void
Output:	An integer ID of an event.
Description:	Returns the most recent event to display if an event is not currently selected.

Name:	String loadMatches()
Input:	void
Output:	A String, formatted as a JSON Array, containing all of the match data for the selected event.
Description:	Queries the database for match data for all matches at the currently selected event needed to populate the Select Match drop down menu on the View Match page of the Group By Event tab.

Name:	String getOverviewTable()

Input:	void
Output:	A String, formatted as a JSON Array, containing the autonomous, teleoperated, and climb point totals for each team attending the selected event.
Description:	Queries the database for event data, then peturns a JSON Array containing the details of the currently selected event needed to populate the ExtJS Event Overview table on the Event Overview table of the Group by Event tab.

Name:	String getOverviewChart()
Input:	void
Output:	A String, formatted as a JSON Array containing the autonomous, teleoperated, and climb point totals for each team attending the selected event.
Description:	Queries the database for match data, then returns a JSON Array containing the details of the currently selected event needed to populate the ExtJS bar chart on the Event Overview table of the Group by Event tab.

Name:	String getRedTeamTable()
Input:	void
Output:	A String, formatted as a JSON Array, containing the autonomous, teleoperated, and climb point totals for each team on the red alliance in the selected match at the selected event.
Description:	Queries the database for match data, then returns a JSON Array containing the details of each team in the given match at the selected event needed to populate the Red Alliance table on the View Match page of the Group by Event tab.

Name:	String getBlueTeamTable()
Input:	void
Output:	A JSON String containing the autonomous, teleoperated, and climb point totals for each team on the blue alliance in the selected match at the selected event.
Description:	Queries the database for match data, then returns a JSON Array containing the details of each team in the given match at the selected event needed to populate the Blue Alliance table on the

	View Match page of the Group by Event tab.

Name:	String generateColorTeamTable(String color)
Input:	A String representing the color of the teams in the selected match to query.
Output:	A String, formatted as a JSON Array, containing the autonomous, teleoperated, and climb point totals for each team on the alliance of the given color in the selected match at the selected event.
Description:	Queries the database for match data, then returns a JSON Array containing the details of each team on the designated alliance in the given match at the selected event needed to populate the designated alliance table on the View Match page of the Group by Event tab.

Name:	int getMatchSearchResult(String search)
Input:	A String containing the match number to search for.
Output:	A positive integer if matching results were found, -1 otherwise.
Description:	Queries the database to see whether or not a match record for a given match exists for the selected event.

Name:	int getTeamSearchResult(String search)
Input:	A String containing the team number to search for.
Output:	A positive integer if matching results were found, -1 otherwise.
Description:	Queries the databse to see whether or not match records exist for a given team at the selected event.

Name:	String getTeamData()
Input:	void
Output:	A JSON String containing the autonomous, teleoperated, and climb, and total point totals for each match for the selected team at the selected event.
Description:	Queries the database for team data, then returns a JSON Array containing the match record details of the selected team need to

	populate the Match Record table on the View Team page of the Group by Event tab.

Name:	String getTeamPieChart()
Input:	void
Output:	A JSON String containing the autonomous, teleoperated, and climb point totals for the selected team at the selected event.
Description:	Queries the database for team data, then returns a JSON Array containing the details of the currently selected team needed to populate the ExtJS pie chart on the View Team page of the Group by Event tab.

Name:	int getSelectedEvent()
Input:	void
Output:	An integer representing the ID of the selected event.
Description:	Returns the identifier for the currently selected event.

Name:	void setSelectedEvent(int event)
Input:	An integer representing the ID of the selected event.
Output:	void
Description:	Sets the value of the currently selected event to the provided ID.

Name:	int getSelectedTeam()
Input:	void
Output:	An integer representing the ID of the selected team.
Description:	Returns the identifier for the currently selected team.

Name:	void setSelectedTeam(int team)
Input:	An integer representing the ID of the selected team.
Output:	void
Description:	Sets the value of the currently selected team to the provided ID.

Name:	int getSelectedMatch()
Input:	void
Output:	An integer representing the ID of the selected match.
Description:	Returns the identifier for the currently selected match.

Name:	void setSelectedMatch(int match)
Input:	An integer representing the ID of the selected match.
Output:	void
Description:	Sets the value of the currently selected match to the provided ID.

3.2.8 GroupByTeamBean

The primary purpose of GroupByTeamBean is to query the database for data related to a selected team.

3.2.8.1 Attributes

Name	Type	Description
selectedTeam	Integer	The team number for the currently selected team.
conn	Connection	A Database Connection
dbconn	DBConnection	Specifies which database is to be connected to

3.2.8.2 Methods

Name:	GroupByTeamBean()
Input:	void
Output:	A new GroupByTeamBean object.
Description:	Constructs a new GroupByTeamBean object with default

	attributes.

Name:	String getEventAveragesTable()
Input:	void
Output:	A JSON string containing the average autonomous, teleoperated, climb, and total points earned by the selected team at each of their events.
Description:	Queries the database for event data, then returns a JSON Array containing the details of the currently selected team needed to populate the ExtJS Event Averages table on the Team Profile page of the Group By Team tab.

Name:	String getEventsLineChart()
Input:	void
Output:	A JSON string containing the total points a team scored in each of their matches for every event they attended.
Description:	Queries the database for event data, then returns a JSON Array containing the details of the currently selected team needed to populate the ExtJS Line Chart on the Team Profile page of the Group By Team tab.

Name:	String getTeamRadarChart()
Input:	void
Output:	A JSON string containing the total points a team scored in the autonomous, teleoperated, pyramid goal, and climb categories over all matches at all of their events.
Description:	Queries the database for match data, then returns a JSON Array containing the details of the currently selected team needed to populate the ExtJS Radar Chart on the Team Profile page of the Group By Team tab.

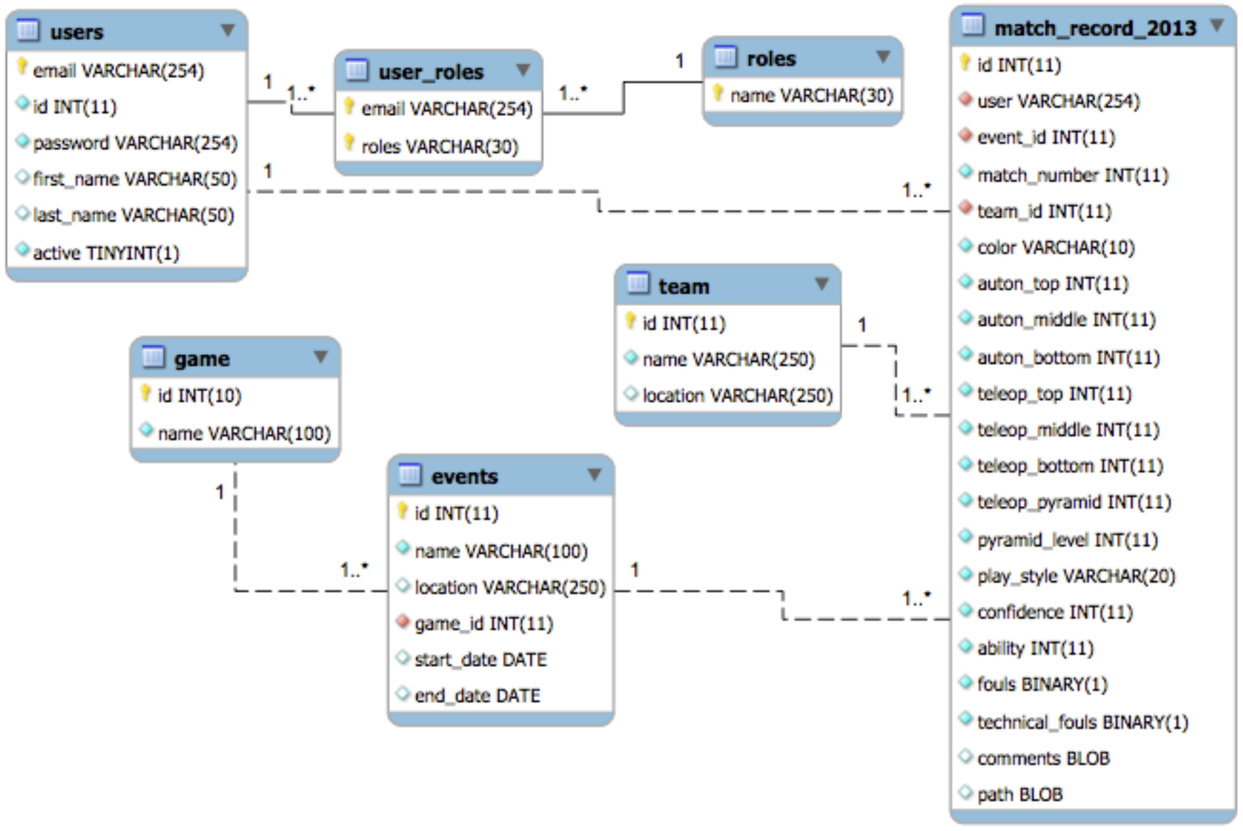
Name:	String getTeamPicture()
Input:	void
Output:	A string containing the URI of the team’s profile image.
Description:	Retrieves the location of a team’s profile image.

Name:	String getTeamMatchesTable()
Input:	void
Output:	A JSON string containing the total points a team scored in the autonomous, teleoperated, pyramid goal, and climb categories over all matches at all of their events.
Description:	Queries the database for match data, then returns a JSON Array containing all of the details for the currently selected team needed to populate the table on the View Matches page of the Group By Team tab.

Name:	int getSelectedTeam()
Input:	void
Output:	An integer representing the ID of the selected team.
Description:	Returns the identifier for the currently selected team.

Name:	void setSelectedTeam(int team)
Input:	An integer representing the ID of the selected team.
Output:	void
Description:	Sets the value of the currently selected team to the provided ID.

3.3 Database Schema



4. Detailed Architecture Diagram

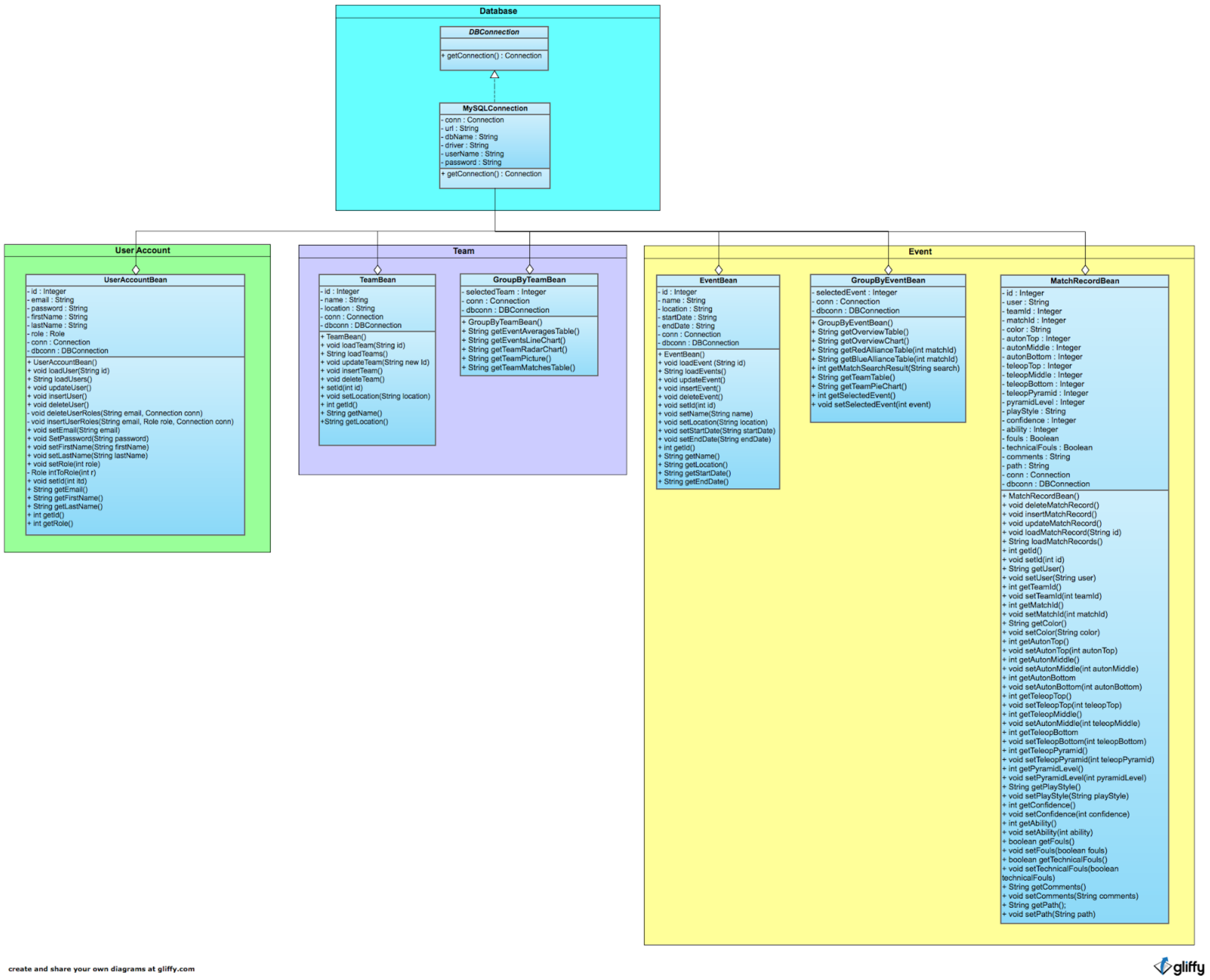


Figure 4. *The detailed system architecture diagram for FRC Scout shows which classes belong to which modules as represented in the component diagram.*

5. Deployment Model

The deployment of the FRC Scout system consists of three major parts: the user's machine, an application server, and the database server. The user's interface to the system is through a web browser on their machine which talks to the application server over HTTP. The application server serves the web pages over HTTP and runs the Java code that communicates to the database via JDBC. The database houses all of the information relevant to the FRC Scout system. For simplicity the application server and the database server may be run on the same machine.

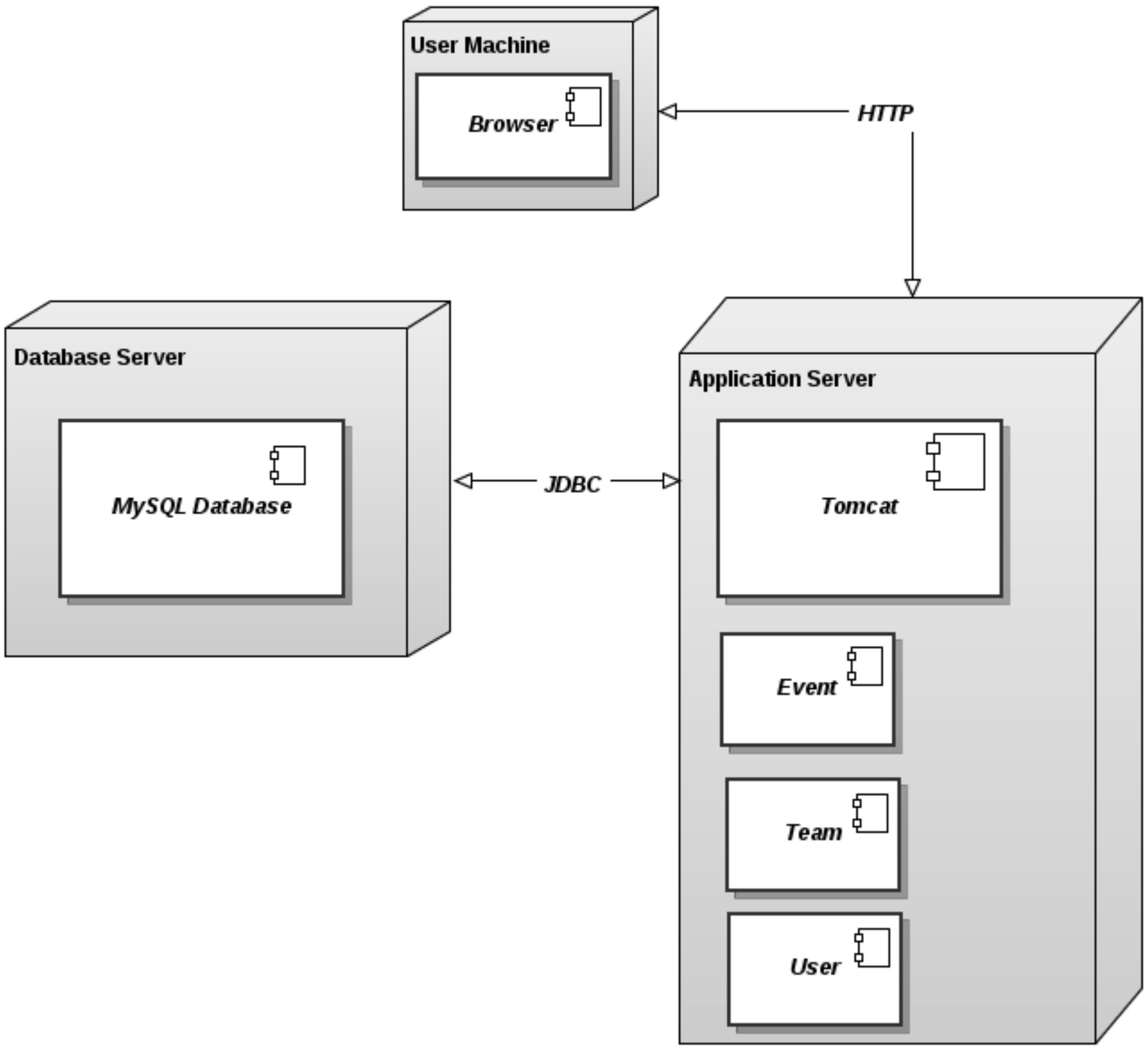


Figure 5. *Deployment Diagram*