

Campus Buddy

by

Anamika Nupur Choudhary

B. Tech., Jawaharlal Nehru Technological University, India, 2010

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2017

Approved by:

Major Professor
Daniel Andresen

Copyright

© Anamika Nupur Choudhary 2017.

Abstract

New to K-State??? No worries!!! This app will be your first friend and help you with everything you may need. Every new incoming student to K-State has to do a set of mandatory activities before they start their classes. Many times they use a pamphlet or word of mouth by students or faculty around, on what to do and whom to visit. But, this information may not be reliable or could have been expired/updated, and students miss on certain crucial things which delays their work. The same follows with various events organized by the college to welcome new students.

This app will be a solution for all these problems. It will provide students with all the details they need before they actually become familiar with the school and even after that. Each student gets to see the To-Do's which are a set of mandatory activities, he/she has to do before they enroll for the classes and also, he can know about various events happening around the university. Students can also suggest new events if they are not already updated in the events list. All these activities are monitored and controlled by the Admin.

Table of Contents

List of Figures	vi
List of Tables	viii
Acknowledgements	ix
Chapter 1 - Introduction.....	1
1.1 Project Description	1
1.2 Motivation.....	2
1.3 Intended Users	3
Chapter 2 - Background and Technologies used	4
2.1 AngularJS.....	4
2.2 jQuery	9
2.3 Springs	10
2.4 HTML	13
2.5 CSS	14
2.6 MySQL	14
Chapter 3 - Related Work	15
3.1 Existing System	15
3.2 Proposed System.....	15
Chapter 4 - Requirements Analysis	17
4.1 Requirement Gathering.....	17
4.1.1 Functional Requirements	17
4.1.2 Non-functional Requirements	18
4.2 Requirement Specification to run the application.....	18
4.2.1 Software Requirements	18
4.2.2 Hardware Requirements.....	18
Chapter 5 - System Design	19
5.1 Class Diagram.....	19
5.2 Use case Diagram	22
5.2.1 Functions of Admin	23

5.2.2 Functions of student	24
5.3 Activity Diagram	24
Chapter 6 - Implementation	26
6.1 Basic Implementation	26
6.1.1 Student Module	26
6.1.2 Admin Module	38
Chapter 7 - Database Design.....	44
7.1 Database Tables	44
7.2 ER Diagram	46
Chapter 8 - Testing.....	48
8.4 Performance Testing	55
Chapter 9 - Security	56
9.1 Security issues in web application and their handling methods	57
Chapter 10 - Conclusion	59
Chapter 11 - Future Work	60
Bibliography	61

List of Figures

Figure 2.1 AngularJS architecture ^[1] (Vardi, n.d.).....	5
Figure 2.2 AngularJS validation 1 ^[3] (blog.angular-university.io, n.d.)	6
Figure 2.3 AngularJS validation 2 ^[4] (Bernado, n.d.)	6
Figure 2.4 MVC Architecture	8
Figure 2.5 Angular MVC Architecture	9
Figure 2.6 Spring Framework ^[8] (www.javatpoint.com, n.d.)	10
Figure 2.7 Spring MVC	12
Figure 5.1 Class diagram 1	20
Figure 5.2 Class diagram 2	21
Figure 5.3 Use case diagram.....	23
Figure 5.4 Activity diagram.....	25
Figure 6.1 Home Page.....	27
Figure 6.2 Registration page when provided wrong password.....	27
Figure 6.3 Registering with KSU email id.....	28
Figure 6.4 Registration page shows message “registered”	28
Figure 6.5 User inactive in registration table.....	29
Figure 6.6 Trying to login without verification	29
Figure 6.7 Verification code sent to KSU email id.....	30
Figure 6.8 Status updated to “Active” in registration table	30
Figure 6.9 Message shown after verification of email.....	31
Figure 6.10 Students home page.....	31
Figure 6.11 Student account info	32
Figure 6.12 Student details which are stored in registration table	32
Figure 6.13 Student To-Do page.....	33
Figure 6.14 Student To-Do’s not completed.....	33
Figure 6.15 Student To-Do’s details	34
Figure 6.16 Map location.....	34
Figure 6.17 To-Do moved to Completed list.....	35

Figure 6.18 To-Do page with no option to select completed or not completed after moving to-do to the completed list	35
Figure 6.19 Events list	36
Figure 6.20 Event details	36
Figure 6.21 Status after student selects interested for an event	37
Figure 6.22 Interested people count increases	37
Figure 6.23 Student adds new events.....	38
Figure 6.24 Event added in sevents table.....	38
Figure 6.25 Admin home page.....	39
Figure 6.26 Students list	39
Figure 6.27 Admin side To-Do page	40
Figure 6.28 Admin adds new To-Do	41
Figure 6.29 To-Do added in todo table.....	41
Figure 6.30 Admin events page	42
Figure 6.31 Admin adds new event	42
Figure 6.32 Admin accept or rejects student event add request	43
Figure 7.1 registration table	44
Figure 7.2 events table	45
Figure 7.3 eventsint table.....	45
Figure 7.4 sevents table.....	45
Figure 7.5 todo table	46
Figure 7.6 todocomp table	46
Figure 7.7 ER diagram.....	47
Figure 8.1 Testing Levels	49

List of Tables

Table 8-1 Testing on Student Module	52
Table 8-2 Testing on Admin Module.....	55
Table 8-3 Performance Observation	56

Acknowledgements

I would like to express my sincere gratitude to my Major Professor, Dr. Daniel Andresen for allowing me to work on this idea, providing me constant encouragement and trusting my abilities to complete this project on time.

I take immense pleasure in extending my heartfelt thanks to my committee members Dr. Mitchell Neilsen and Dr. Torben Amtoft for their encouragement and for taking the time to serve on my committee.

I also would like to acknowledge the help and constant support provided by the academic staff of the Department of Computing and Information Sciences, when and where it was required.

Finally, I would also like to thank parents, siblings and friends to motivate me all throughout this project. I wouldn't have finished this without their constant support.

Chapter 1 - Introduction

1.1 Project Description

Campus Buddy is a web application which would help K-State students, with everything they would need before they start their classes. Currently there is no exclusive app (mobile or web) which would help students to manage their To-Do's for the first week check-ins. Hence, students have a hard time trying to figure out what they should do, whom they should see for help and which department they should visit for the check-ins. These to-dos' can be anything from collecting a new id card to verifying your documents at the grad school. Currently students get few details from student center or by communicating with other fellow batch mates. But the information they get is not always an updated one. Also, as students are new to the school they may not know all the routes in the campus and keep roaming around which may delay their work. Also, in the first week of school, K-State organizes many events to welcome new students. These events can be anything like introduction session, ice-cream social, coffee hour, campus tour etc., For few of these events, they get email invitation from K-State, receive brochures but in most of cases they end up going by word of mouth from other students. New students may not know the routes to the location where the event is planned, can always miss the event invite in the huge clutter of other ones and may lose the brochures, giving a miss to the opportunity which would be helpful for them in the future. To avoid these problems and help students with managing all these things this application would be really useful. It is one place where a new student can find all necessary details. The admin updates all the events and to-dos from time to time to give exact information to students. This app will have the location details, contact person and a map to each place which students can use to reach the destination. There are few to-dos' which can only be performed once the previous ones are completed. For example, student can only enroll for classes

after he completed certificate check. So, this app will also help students to keep track with the sequence of to-do list. The event list will have the event date and location which can be helpful to the students. As all these details are stored in one place, there is no chance that the student will lose any details and miss any event. Apart from this, if a student has any information about new event he can add it as well. He just has to provide the details of the events and the location of the event. And once the admin approves his add request the event is added to the events list. This functionality will be really helpful to admin as he regularly gets update about the events from the users and this will lessen his work.

Apart from helping new students, the main aim of this project is to help me use and learn new technologies. For this project, I am using AngularJS, jQuery, HTML, CSS, Springs and MySQL for the database. I did not have any prior experience working on AngularJS and Springs previously. This project gave me enough exposure to these technologies.

1.2 Motivation

This app was developed out of my personal experience at K-State. When I first visited the school, I did not know any routes neither did I have any friends to help me with anything. I did not have a registered KSU email id as well to receive any important information from the school. I had a lot of trouble finding the routes and many times I kept roaming around the campus without any help. I always thought if I had a friend, who knew every place and had information about every mandatory check-in required, it would be really helpful. But then nobody would actually know all the correct details, as to-do list always gets updated every semester and even if somebody knows all the routes and details, it may not be updated one. Also, we do not always find people with so much free time to accompany us. Keeping all these situations in consideration, I thought an online

friend would be a best bet. You can always trust this app with any details you may need about the events or to-dos'. This app will be very useful to the students like me who are new and uninformed.

1.3 Intended Users

College Students: Every K-State student with a registered K-State email id can access this application. Once the student registers his account, he gets verification email to the registered K-State email account with domain @ksu.edu. Once his account is verified he can access this app. He can see the list of to-do's, events and a functionality to add new events.

Admin: Admin is the sole responsible person to update data in this app. He can add, delete or update new events and list of to-do's. He will also have list of users of the system to keep a track of the system. He can also approve/deny every event add request of the students after careful consideration, and the ones approved are added automatically to the event list.

Chapter 2 - Background and Technologies used

Apart from helping new students, the main aim of these project was also to learn new technologies which I am not versed with. So, I used many latest technologies in my implementation. These technologies are used in various areas according to their necessity and importance. The technologies I used in my implementation are AngularJS, HTML, CSS, jQuery, and Springs and MySQL for my database implementation.

All the validations in the project use AngularJS and for the frontend design and implementation, I used HTML, CSS, jQuery and Springs.

2.1 AngularJS

AngularJS is a JavaScript MVC framework useful while developing single page applications, in short to create dynamic web pages. In Angular data moves inside a single-page application, dynamically updating the view as the data changes without the help of any specific listener code. It is used to build complex and dynamic features quickly with simple and declarative templates using existing components. Angular provides validation at client side rather than server side validation and it is capable to enhancing the functionality of HTML, CSS and JavaScript.

AngularJS supports the following web model:

Server - Provides the client with a set of initial HTML data, then for future requests, it just returns JSON data.

Client - Takes in that JSON data, and updates the HTML dynamically.

2.1.1 AngularJS architecture:

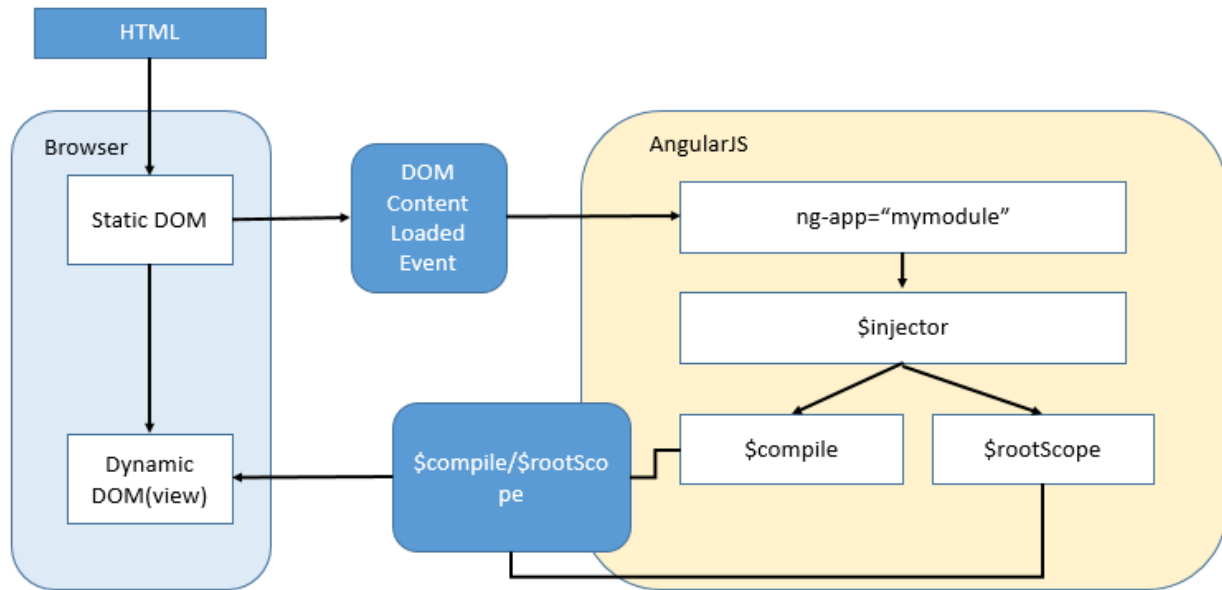


Figure 2.1 AngularJS architecture^[1] (Vardi, n.d.)

Below is the basic architecture description:

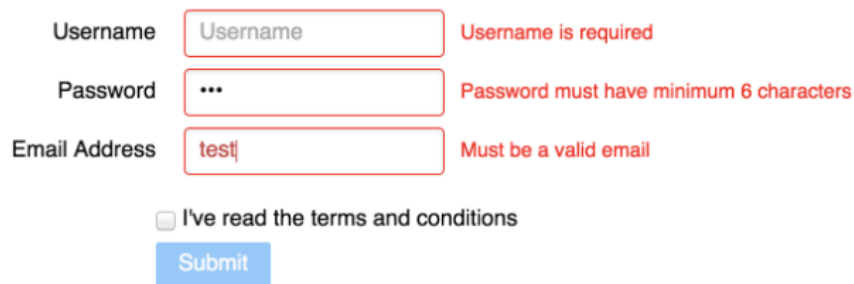
- ➔ Angular bootstrap is created.
- ➔ A module “mymodule” is created.
- ➔ \$injector is then created and configured to the module “mymodule” and then the object is then retrieved from the injector by name.
- ➔ When \$injector is done retrieving all the objects, we execute the code.
- ➔ We run compile() function when all static HTML pages are parsed into DOM and then the link() function is run.

2.1.2 Validation using AngularJS^[2] (Smith, n.d.)

Validating web pages using AngularJS would be advantageous as it provides better user experience than a server-side validation, as it uses client-side validation and the user gets instant feedback on how to correct the error. It monitors the input entered by the user and notifies him about the current state of the form. Also, it keeps track of all the details such as which fields are touched or modified in the form.

Few highlights of validation using AngularJS

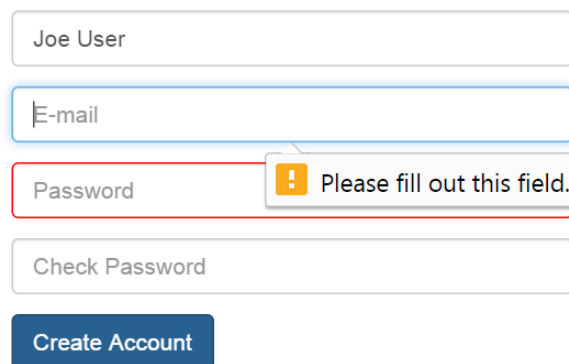
- highlights fields in error dynamically as we type.
- provide inline messages while the user is typing in a field.
- disable the submit button until all the needed data is available and the terms and conditions checkbox is checked.



A screenshot of a form with three input fields: Username, Password, and Email Address. Each field has a red border and a red error message to its right. The Username field contains the text 'Username' and the error message is 'Username is required'. The Password field contains three dots '...' and the error message is 'Password must have minimum 6 characters'. The Email Address field contains the text 'test' and the error message is 'Must be a valid email'. Below the fields is a checkbox labeled 'I've read the terms and conditions' which is unchecked. At the bottom is a blue 'Submit' button.

Figure 2.2 AngularJS validation 1^[3] (blog.angular-university.io, n.d.)

The image below shows the form in action, after entering some data and pressing the submit button.



A screenshot of a form with four input fields: 'Joe User', 'E-mail', 'Password', and 'Check Password'. The 'Password' field has a red border and a tooltip with a yellow warning icon and the text 'Please fill out this field.' pointing to it. Below the fields is a blue 'Create Account' button.

Figure 2.3 AngularJS validation 2^[4] (Bernado, n.d.)

In normal JavaScript based validation page, we do not have these features. Many times we click submit button without entering all the fields and this gives error message. In secure and critical applications like the one used for banking, if there is an error in the page, all the fields are updated

and we have to redo the whole process from the beginning, wasting crucial time. Also, by providing inline and dynamic error messages, Angular gives us the scope to update the field as we continue to the next one without having to come back to the application and redo the whole thing. But, this has a limitation; there is no way to inform user that password and check password should be the same. In such cases we create our own custom validation functions using directives.

Few Angular properties:

- ➔ \$pristine - No fields have been modified yet
- ➔ \$dirty - One or more have been modified
- ➔ \$invalid - The form content is not valid
- ➔ \$valid - The form content is valid
- ➔ \$submitted - The form is submitted
- ➔ ng-untouched - The field has not been touched yet
- ➔ ng-touched - The field has been touched

2.1.3 Features of AngularJS and its advantages over JavaScript^[5] (Ruebbelke, n.d.)

- 1) **Two-way data binding:** It is automatic synchronization of data between your view and model. Whenever a value in the model is updated, the view replicates it automatically and vice versa. This feature is not available in JavaScript and hence the development becomes really fast.
- 2) **Single page application:** All the user needs are put in one page without the need to move back and forth between different pages, which is really confusing. Instead, the content is loaded asynchronously on the same page and just the URL is changed to reflect the selection. This supports Routing.

- 3) **Better Template:** It is just plain HTML page with extra capabilities. A directive in AngularJS makes HTML do new things, by adding new elements to it which is not possible with plain HTML syntax.
- 4) **Easy collaboration:** Collaboration between developers become really easy as they can work independently of developing the UI in HTML and use declarative binding syntax to bind all the different UI components developed by various developers in one single data model with ease. This also promotes modular development.
- 5) **Dependency Injection:** Due to this feature in AngularJS testing becomes a cake walk. We can just ask for the dependencies rather than having to search for them.

2.1.4 MVC architecture

MVC is a software design pattern for developing web applications. It has 3 important parts:

Model – It is responsible for maintaining data.

View – It is responsible for displaying the data to the user.

Controller – It controls the interactions between the Model and View.

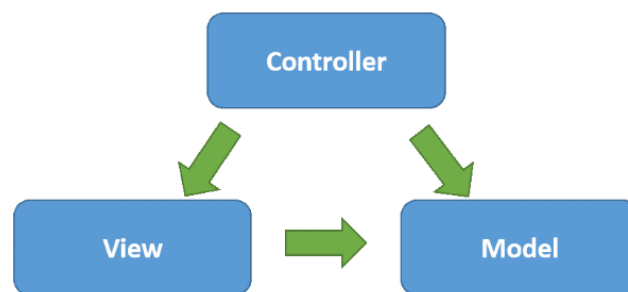


Figure 2.4 MVC Architecture

MVC distinguishes the application logic from the user view. The controller receives all the requests sent to the application and then works in integration with the model to displays the generated output to the user requesting the service.

2.1.5 Angular MVC Architecture ^[6] (www.Pluralsight.com, n.d.)

MVC in angular is implemented in JavaScript and HTML. For the view, we use HTML, and for the model and controller we use JavaScript.

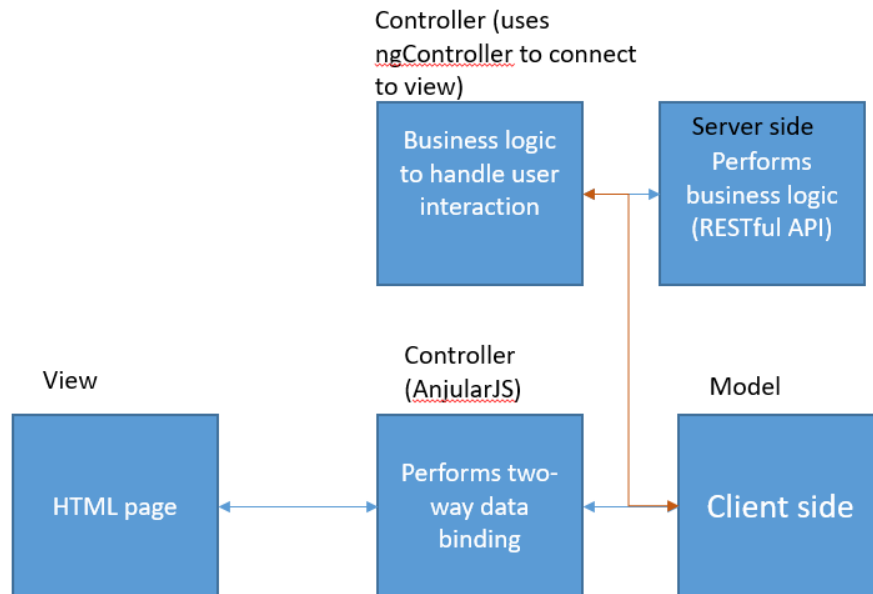


Figure 2.5 Angular MVC Architecture

Model: It contains the data to be displayed, data collected as the input and functions invoked by the user. Generally, we can directly use `$scope` as a model.

View: It is a HTML page with reference to AngularJS framework to include bootstrap and directives to manipulate the DOM.

Controller: For this purpose, we use `ng-controller` directive. It will be used for the handling and manipulating all the data behind the UI.

2.2 jQuery

jQuery is a JavaScript Library. It makes coding really simple as many lines of a JavaScript code can be written into a single line. It also makes Ajax, data manipulation and traversal, animation

etc., really simple. Using jQuery tools, we can communicate to the server without reloading the whole page. In jQuery we select HTML elements and perform operation on them using action() function. In jQuery like any other scripting languages \$ hold a lot of value. \$ sign is used to define or access jQuery. For selecting HTML elements, we use selectors with the element name.

2.3 Springs

2.3.1 Spring Framework

It is Java platform providing java applications a platform and comprehensive infrastructure for their development. Spring Framework helping in the development of enterprise applications in particular using POJOs.

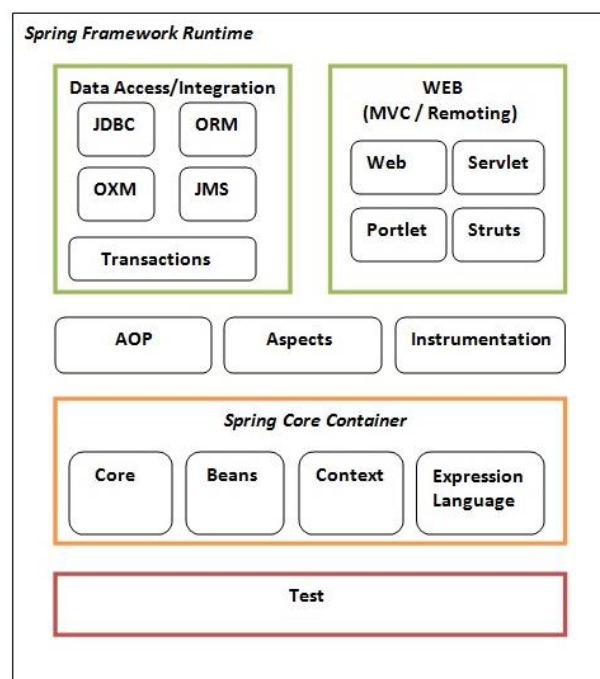


Figure 2.6 Spring Framework^[8] (www.javatpoint.com, n.d.)

Description:

Spring Core Container: It is basis for the complete Spring framework. It is used in all the other modules. This module injects dependencies, so that we need not use factory classes and methods.

- ➔ Core: It provides IOC and Dependency Injection features
- ➔ Beans: It provides BeanFactory for implementing factory pattern.
- ➔ Context: It supports EJB, JMS and Basic Remote features.
- ➔ Expression Language: It supports multiple things like named variables, logical and arithmetic operators and many others.

AOP: It stand for Aspect Oriented Programming. It is implemented in Java based on AOP Alliance API. This allows integration of existing AOP alliance complaint to the spring or allows migration of the component implemented using Spring AOP to other AOP.

Data Access/Integration ^[9](htt1):

- ➔ JDBC: In normal JDBC API we end up writing lot of code, for creating connection, handle transaction, exception handling. This complex coding is handled using JDBC in Spring API which will provide JDBC-abstraction layer.
- ➔ ORM: Using ORM, we have to write less code for the database connection, It provides abstraction for object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
- ➔ OXM: This module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
- ➔ The Java Messaging Service JMS has features for creating and using messages.
- ➔ The Transaction module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web Module:

It helps in the development of web application in a simple way. It also supports MVC based application development.

2.3.2 Advantages of Spring Framework

- ➔ It has predefined templates.
- ➔ Springs doesn't require server to run. By using POJOs, we need not use application server but we can use Tomcat or other such servers.
- ➔ Spring framework has a well-designed MVC framework, which serves as great alternative to web framework available.
- ➔ In Springs we do not start everything from scratch, rather it uses the existing technologies like ORM frameworks, JEE and many others. In short it is versatile.
- ➔ Testing becomes a really easy, as Springs use POJOs which in turn uses dependency injection for injecting test data.
- ➔ Springs work in modular fashion, where no module is dependent of each other.

2.3.2 Spring MVC^[10] (Rajput, n.d.)

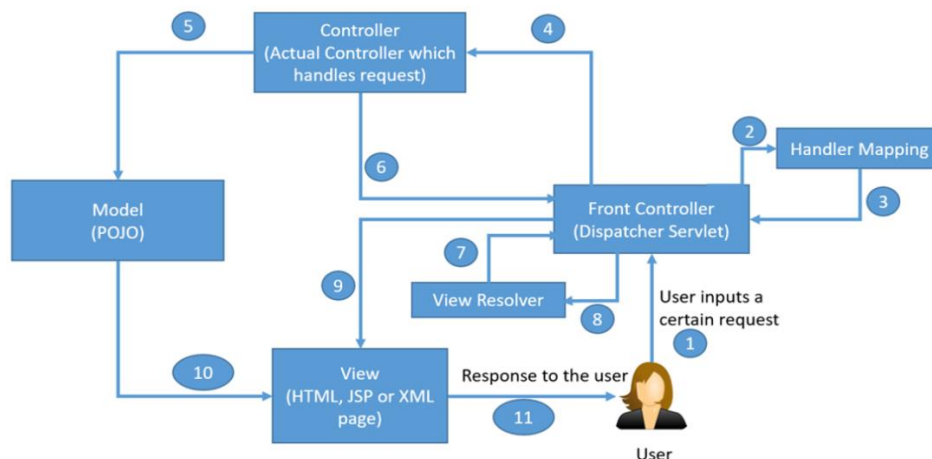


Figure 2.7 Spring MVC

Step 1: Used sends the request to the Front Controller (DispatcherServlet).

Step 2: DispatcherServlet dispatches to the HandlerMapping. HandlerMapping selects the corresponding controller for the request URL given by the user.

Step 3: HandlerMapping sends back the respective controller information and selected Handler to the DispatcherServlet.

Step 4: The DispatcherServlet then selects the controller to perform business logic.

Step 5: The Controller then performs all the business logic and sets the processed result in Model.

Step 6: Once all the process has finished the Dispatcher Servlet is updated.

Step 7: DispatcherServlet dispatches the task of resolving the View corresponding to the View name to ViewResolver.

Step 8: ViewResolver provides the View mapped to View name to the DispatcherServlet

Step 9: DispatcherServlet dispatches the provided information of process to the View.

Step 10: View also receives the model data from Model

Step 11: It then provided the response to the user (person requesting the service).

Advantages of Spring MVC:

- ➔ More annotation based, i.e., reducing the extensive use of configuration and metadata.
- ➔ It supports different views (JSP, XML, PDF etc.,) and MVC frameworks.
- ➔ Supports RESTful URLs.

2.4 HTML

It stands for Hyper Text Markup Language. It is used to create static web pages. It is one the basic languages to learn static web designing.

Advantages of HTML:

- ➔ It is simple to code.
- ➔ It is easy to learn.

- ➔ Widely used.
- ➔ Can be used to integrate many languages.
- ➔ It is very flexible.
- ➔ It provides various templates which makes designing easy.

2.5 CSS

It stands for Cascading Style Sheets. It is used to beautify the web pages. It allows separating content of an html document from the style and layout of that document. CSS allows developer to create stylish websites and make them look attractive. In CSS, just by changing the style, the elements in the web page is updated automatically. CSS is also much faster than a plain HTML code.

2.6 MySQL

MySQL is a database management system and is open source relational database. To store all the information provided by the Admin and Student details are stored in database. All the information in MySQL database is stored in tables. The front-end of the project is connected to backend MySQL. When the user requests data from the front-end, it retrieves the data from the MySQL tables. MySQL is scalable and has better performance than many other relational databases. Using MySQL, we can add, delete and update information in the tables. Each table in a MySQL database has a primary key, which is a unique identifier for each record.

Chapter 3 - Related Work

3.1 Existing System

In the existing system, K-State uses various applications to help the students to manage their courses, their financial details, employment notifications and many such. For example, there is Canvas where students can add and manage course, submit assignments, receive grades, see the course materials and do many other course related stuff. Then we have KSU CES website, where student can get all the information related to career fairs, companies visiting the campus for placements and the company details. Also, we have K-State Online application from where student can access K-State webmail, manage and pay tuition fee, and receive paycheck if he/she is employed on campus. These apps are really helpful for the students. But there is no application which will help students in the initial days of their school. Nobody or no app gives a full proof information about the mandatory things a student should be doing before they enroll and start their classes. Few events details are sent through K-State webmail but they do not have location maps nor do they have other mandatory information like point to contact, their email and contact details (most of the times). Due to all these issues students face a lot of difficulty in the initial days.

3.2 Proposed System

In the proposed system, we are trying to develop an application which will help students in their first few weeks at K-State. This app will provide students with the list of to-do's which will give them the sequence of the steps which they need to fulfill. The To-Do list will have, all to-dos' Every step will have a to-do, with the details of the place and the map to the location. It will also inform student the things they need to carry with them to the location. For example, if the to-do says, they need to check-in with the grad school, it actually means they will have to submit their previous transcripts; so in this case the mandatory documents to carry would be the transcripts. Also it would

provide students with the event list around the campus along with the details of the point of contact. It will have two different type of users; one is the admin, who will update the information for the student (who is the user) and the students. To provide security and restrict the app to just K-State students, every time the user (K-State student) registers to the app, he gets a verification mail to his K-State account. Also, there are chances that the Admin may not know every new event in the university. To provide him some help, there is also an option where students can help him by add the event details, which needs to be re-verified by the Admin. I believe all these capabilities will help the student immensely in his start of journey at K-State, and hence I named it Campus Buddy.

Chapter 4 - Requirements Analysis

4.1 Requirement Gathering

Software development starts by gathering requirements which you may need in your project implementation. It is one of the crucial steps in the development process. Only if we have all the correct requirements beforehand, prior to starting our actual implementation, we will have the expected output. Else, no matter how detailed and extended our design and code is, we will end up with unexpected results in our final output. After detailed analysis of the requirements, I could segregate them into functional and non-functional requirements

4.1.1 Functional Requirements

It describes what system should do.

- ➔ Student registration
- ➔ Admin and Student Login
- ➔ Admin should be able to add to-do list
- ➔ Admin should be able to add events
- ➔ Admin should be able to see the students registered
- ➔ Student should be able to see to-do list added by the admin
- ➔ Student should be able to see the events added by the admin
- ➔ Student should be able to add events.
- ➔ Admin should be able to validate the event added by the user, which will add it to the events list.
- ➔ Student email address should be verified.

4.1.2 Non-functional Requirements

It describes the working of the system. The non-functional requirements such as security and integrity are provided to this system through verification and validation. (i.e., by prompting error messages if anything goes wrong)

4.2 Requirement Specification to run the application

(Note: These are the requirements I used while developing this application)

4.2.1 Software Requirements

Operating System: Windows 10

IDE: Eclipse Luna

Servers: Apache Tomcat 7.0.27, Wamp 2.0

Frameworks and Web Interface: Spring MVC, phpMyAdmin

Database: MySQL

Front End: HTML5, CSS3, JavaScript, jQuery

Browser: Any browser should work (preferably Chrome and Firefox)

4.2.2 Hardware Requirements

Processor: Intel core i5

Processor speed: 2.30 GHz

RAM: 8 GB

Chapter 5 - System Design

System design gives static and dynamic view of the system. It defines various portions of the project like modules, components etc., and their functionality. These are actually laid before development to have a clear idea on the requirements beforehand. System designing is actually done on two separate system models. Static Models, which gives how the static elements in the code like class and objects should look like and dynamic models give the behavior of the static components. Class and object diagram provide the static view of the system where as use case and activity diagram provide the dynamic aspects of the model.

5.1 Class Diagram

It provides the static view of the system. It shows the classes used in the system, their attributes and functions. It doesn't give the functionality or the data flow. It provides information about the things which doesn't change in the system. In this project, there are classes in 4 different section. Controller classes, dao classes, mail class and vo class.

Controller classes: The Controller classes pulls data from the request and passes it to the class which request the service. In short, they handle web requests. In this project there are multiple controllers like the StudentListController which controls the student side activities, AdminDataController which handles Admin data and so on.

VO classes: The data is actually sent to DAO through VO. In this project, the classes in vo are used only to set values to the to-do, events and user(student).

DAO classes: Classes in this portion interact with the database. They generally contain all the query code to connect and interact with database.

Mail Classes: Class in mail, has the methods and attributes to send verification email to KSU email id.

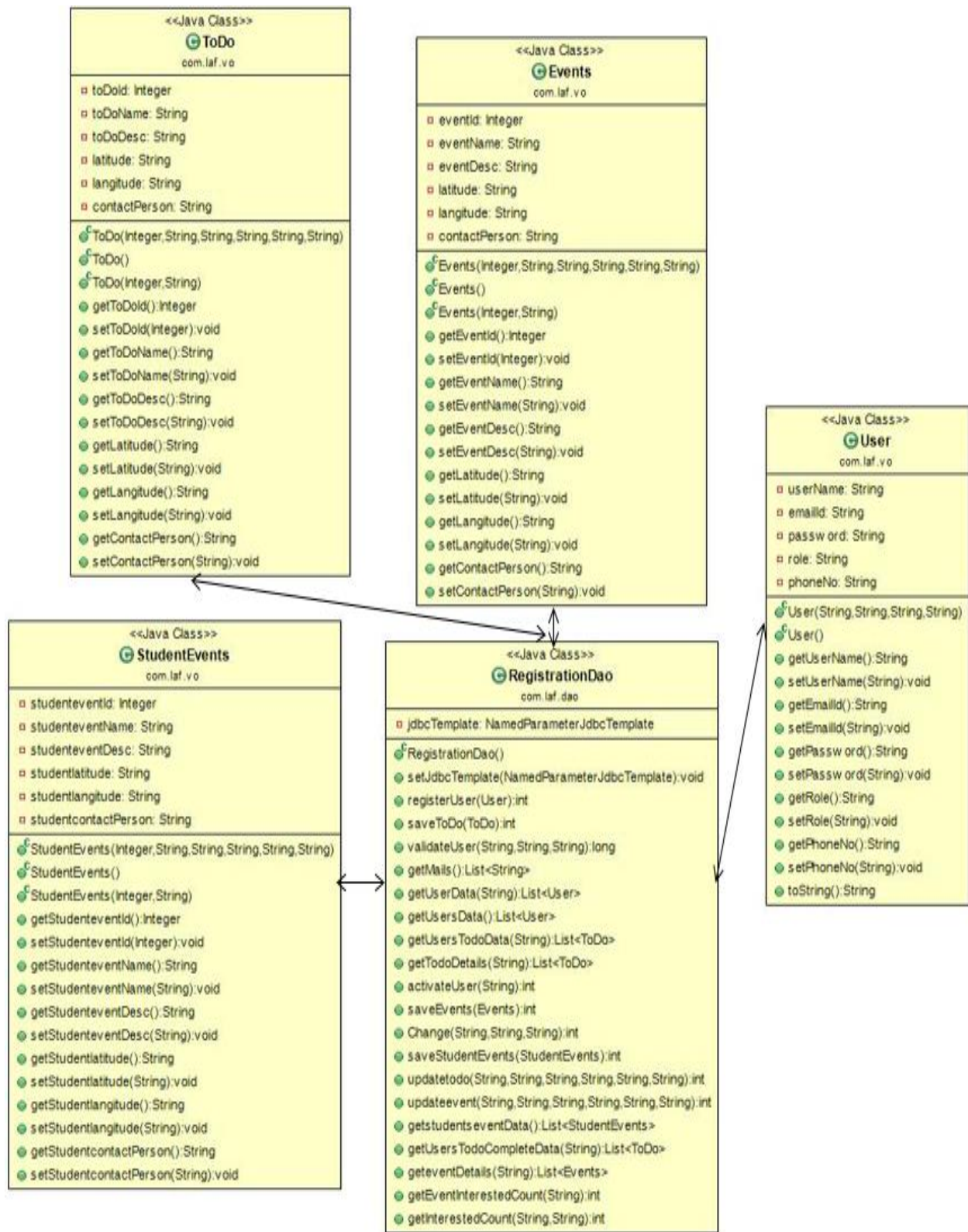


Figure 5.1 Class diagram 1

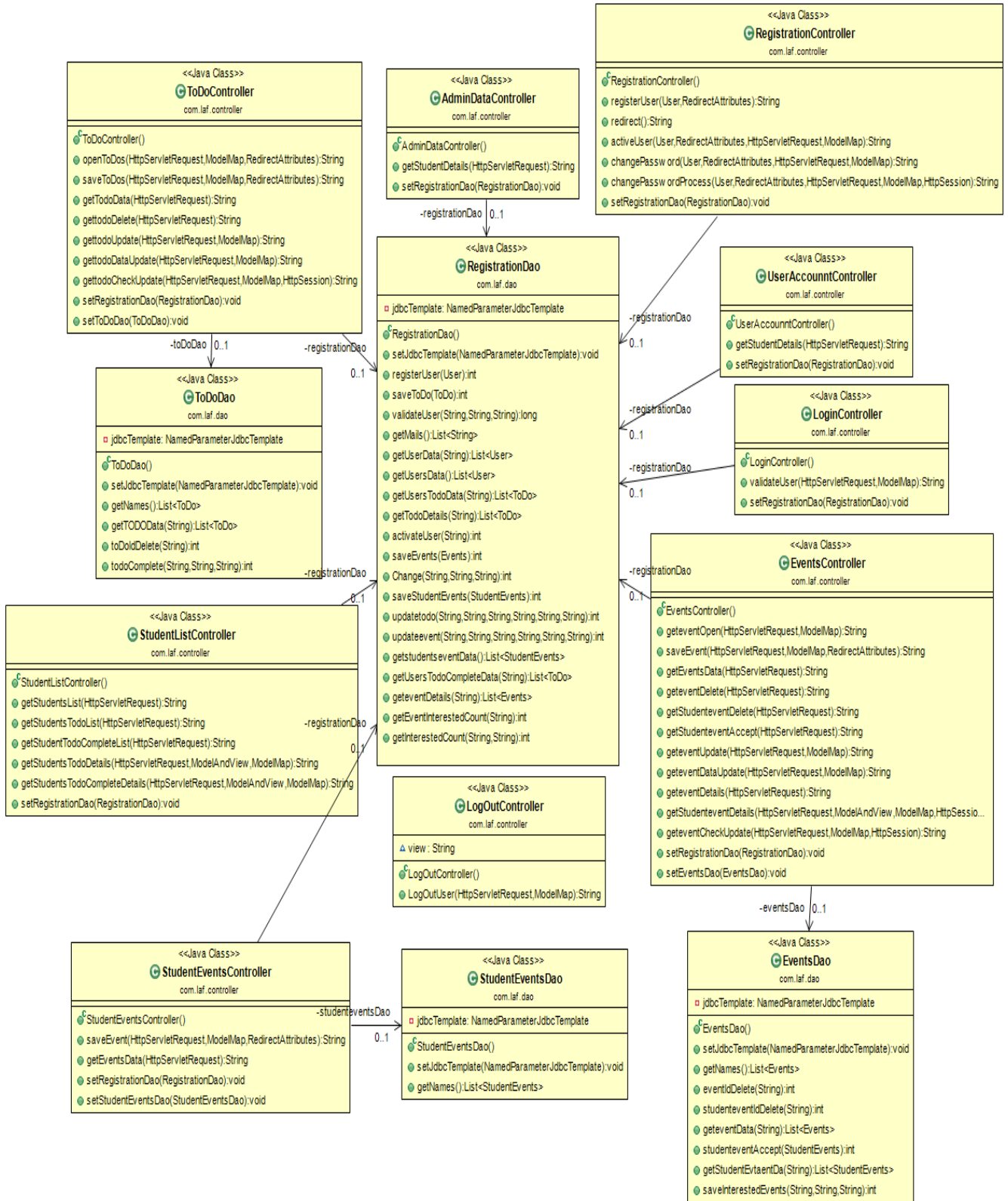


Figure 5.2 Class diagram 2

The class diagram is divided into 2 parts for simplicity. In class diagram 1, there are all model classes and they communicate with RegistrationDao class which has database connections to all important tables. The classes ToDo, Events, StudentEvent and User are used to set the values to the various variables and objects in the database through RegistrationDao and get the values from the RegistrationDao, which in turn communicates with the database to provide values to the model and the model then returns them to the controllers or the view.

In class diagram 2, there are controller classes which communicate with their respective DAO classes. These controller classes are responsible for the views through which the user requested any service. For example, from todo.jsp when a request is raised, dispatcher servlet maps it to ToDoController and this in turn communicates with ToDoDao. ToDoDao gets value from the database that is required by the controller to serve the request. In the figure, there are around 8 controller classes and they communicate with respective Dao. Also, these controllers communicate with RegistrationDao which has all important connections to the database like getting ToDo for respective User, extracting the events which a particular user is interested in and so many other things.

5.2 Use case Diagram

Use-Case diagram shows the dynamic view of the system i.e., the behavior of the system. It doesn't focus on system as such but focuses on users of the system. Use cases help to manage large projects by decomposing it into functions.

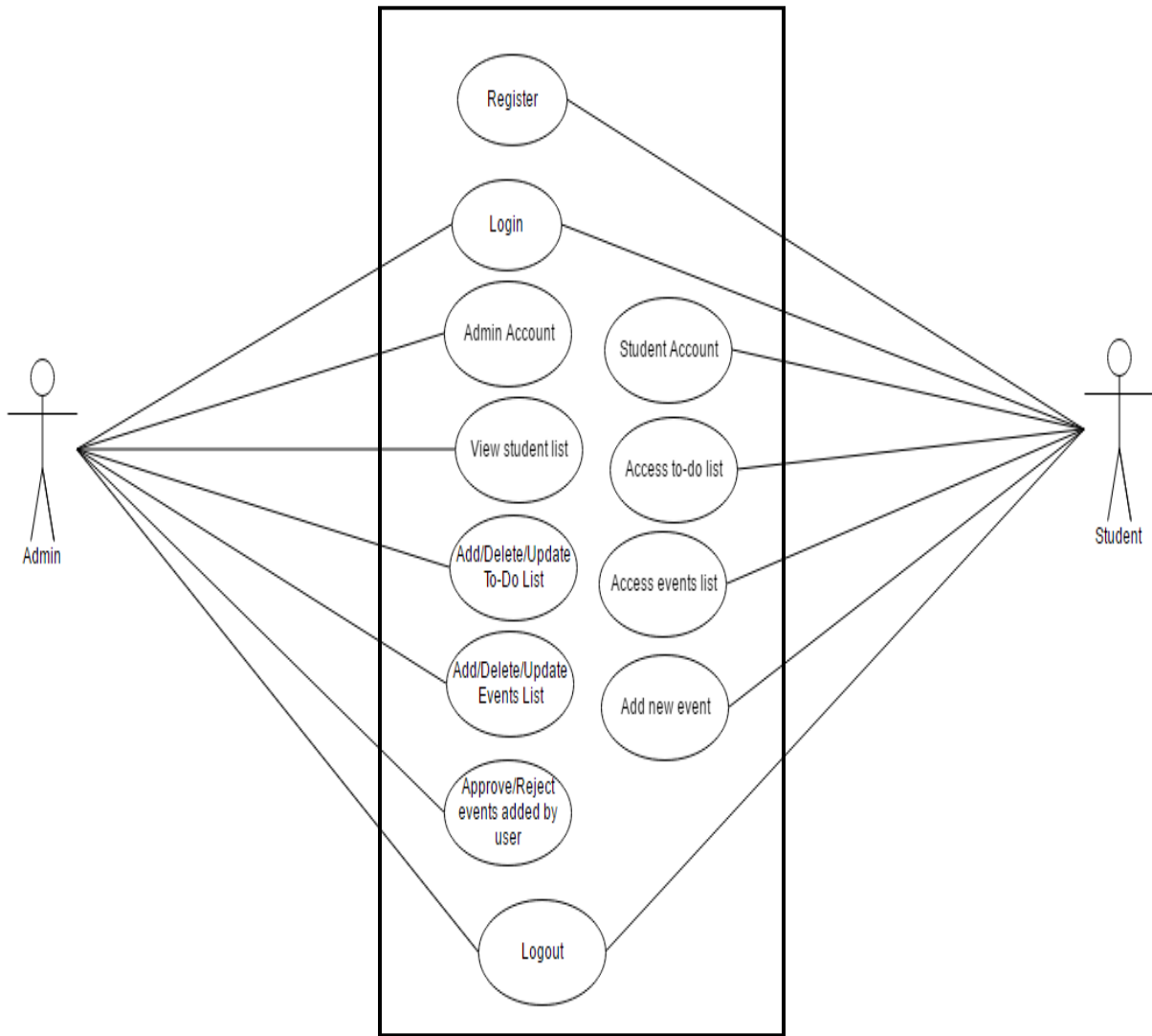


Figure 5.3 Use case diagram

In this project, there are two users. Admin and student.

5.2.1 Functions of Admin

Login: He can login with his credentials.

Admin Account: This function will provide the admin details.

View students list: Admin can see all the students who are already registered.

Add/Delete/Update To-Do List: Admin can add new to-do's and, delete and update previous ones.

Add/Delete/Update Events List: Admin can add new events and, delete and update previous events.

Approve/Reject events added by user: Admin can approve or decline the event added by the student.

Logout: Admin logs out from his account.

5.2.2 Functions of student

Register: Student registers using his KSU email id.

Login: Student can login to his account, using the username and password provided.

Student Account: It will provide account details of student.

Access to-do list: Student can see the to-do list. Here each to-do can be expanded and can be checked as completed or not completed. Once the to-do is completed, it moves to to-do completed list.

Access events list: Student can see the events list. Each event be expanded and can be checked as interested or not interested. Once the to-do is completed, it moves to to-do completed list. This updates the people interested count for this event.

Add new event: Student can add new event with all the necessary details.

Logout: Student can logout from his account.

5.3 Activity Diagram

Activity gives the work flow of execution in the project. We use forks and joins in activity diagram to show how activities flow inside a system. From the picture 5.3, it can easily be inferred that there are two different flows. Admin and student. Student has to register and then login. Once the student is inside the student homepage, he can view his account, To-Do list,

event list and can also add new event. But the admin just can log in directly. Once he is in his homepage, he can add/delete/modify To-Do's, events and can approve students add request.

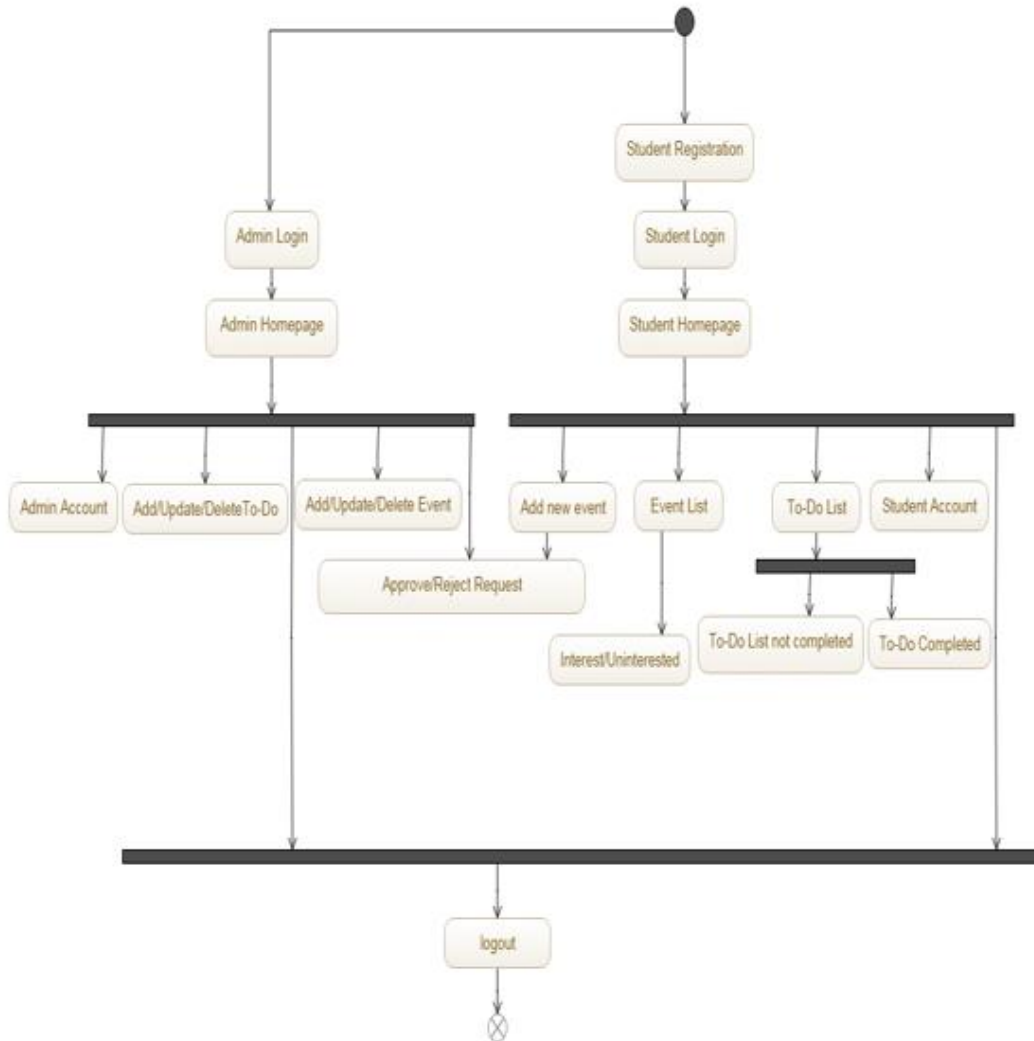


Figure 5.4 Activity diagram

Chapter 6 - Implementation

I have implemented this project in 2 modules. User module and Admin module. The main purpose of this app is to provide students the details of the things they need to do in the first week of their school (list of to-do's) and the list of events around university.

Explanation of the Modules:

Admin Module:

Admin has a special username and password. He can enter to-do list, event list, approve events added by student, see the student list.

User/Student Module:

Students can see the to-do and events list added by the Admin and they can also add new events which needs to be approved.

6.1 Basic Implementation

This project has been implemented in 3 main portions. There is a **controller**, which are the .java files which handles all the requests, which it receives from the .jsp files. We have DAO's controllers which are again .java files. They have all the queries which are required to interact with the database. The DAO classes send back model classes to the Controller class in order to be sent to the view layer. Then we have .jsp pages, which send requests to perform certain functionalities. They also contain the HTML code, to provide application view to the user.

6.1.1 Student Module

Below is the first screen which is displayed on running the application. This is a basic HTML page and is displayed for just few seconds before it is redirected to registration page.



Figure 6.1 Home Page

Registration page in the page from where the actual project flows. Admin can directly login from here using predefined username and password and role as ADMIN. Students who are new should register before being able to use this app. As this application is restricted to the K-State students, if student provides any other email id, except with @ksu.edu extension, it will show error. In the picture 6.2, I tried entering a Gmail id and gives an error.

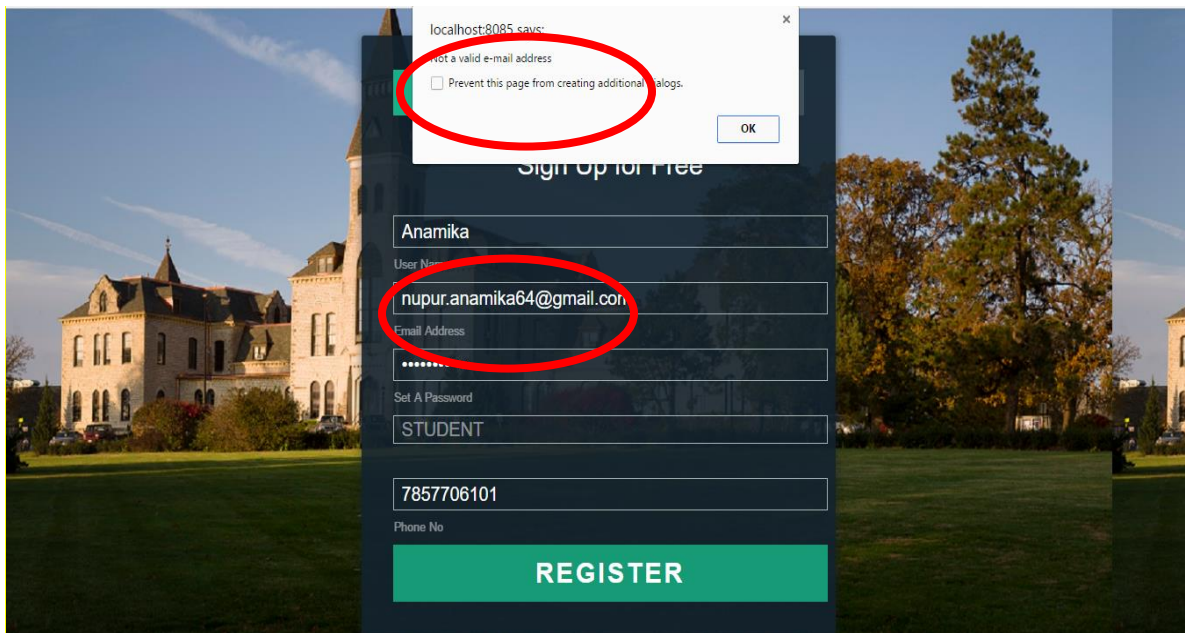


Figure 6.2 Registration page when provided wrong password

Once we enter a valid K-State email id, as in figure 6.3 it allows student to register and gives a message “you have been registered”, as in figure 6.4.

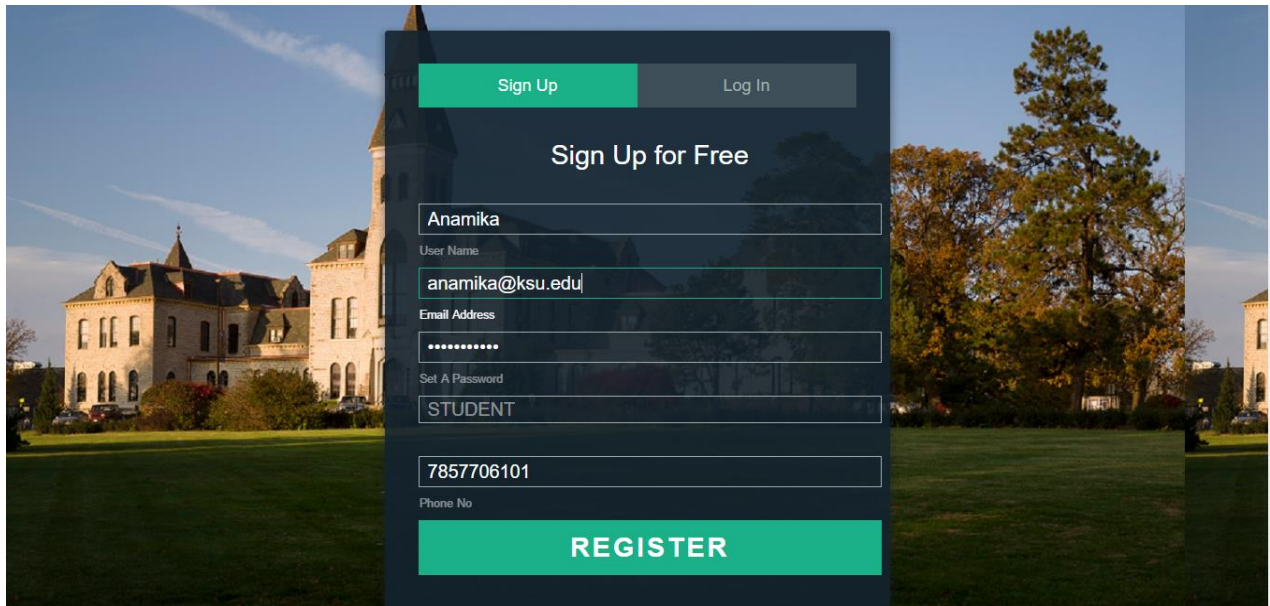


Figure 6.3 Registering with KSU email id

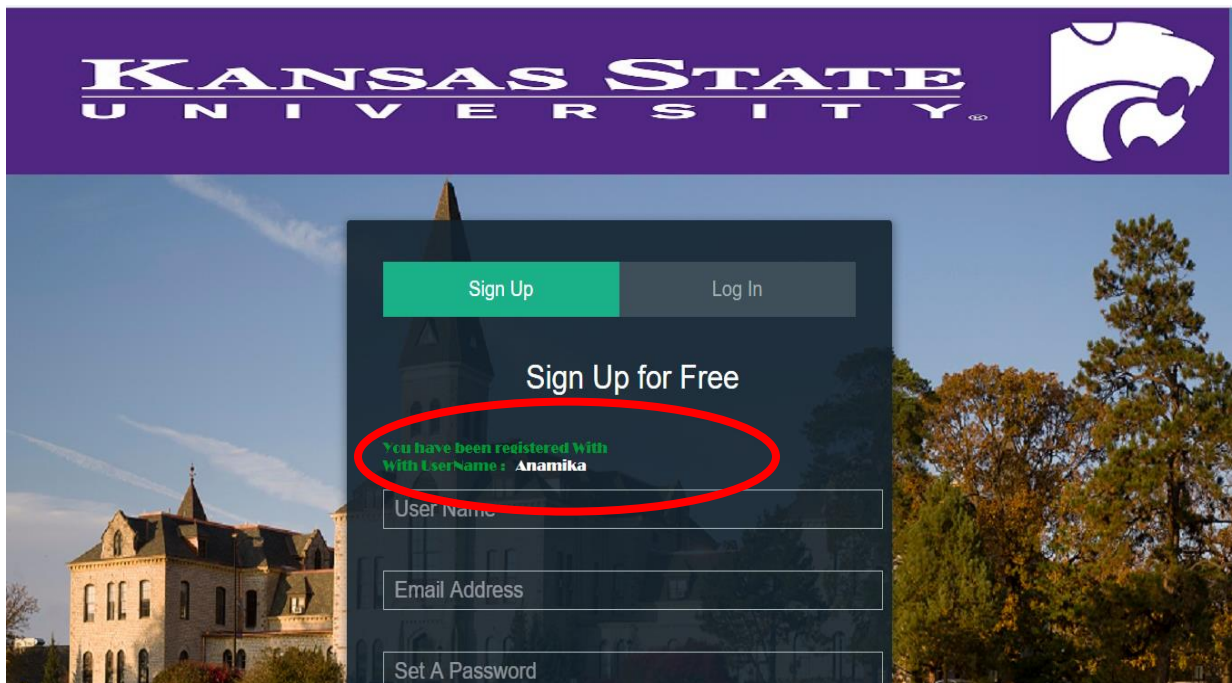


Figure 6.4 Registration page shows message “registered”

At this point the student has just registered but has not verified his email and if he tries to login, it will throw an error as in Figure 6.6. Only after the student confirms that he indeed is a K-

State student, he is allowed to access the app. Figure 6.5 is the snapshot of the database below verifying the email id. It shows user “Anamika” is still inactive.

userName	emailid	password	role	phoneno	status
admin	admin@gmail.com	X+tGwEBkr7aJJSpQI9fUJg==	ADMIN	5655665552	Active
Anamika	anamika@ksu.edu	SHLwDw/NGYA=	STUDENT	7857706101	Inactive
Shravan	shravandg@ksu.edu	qr956UDEjRs=	STUDENT	2486642345	Active
Prathap	prathap93g@gmail.com	b4WE4bXVBj0=	STUDENT	7306827578	Active

Figure 6.5 User inactive in registration table

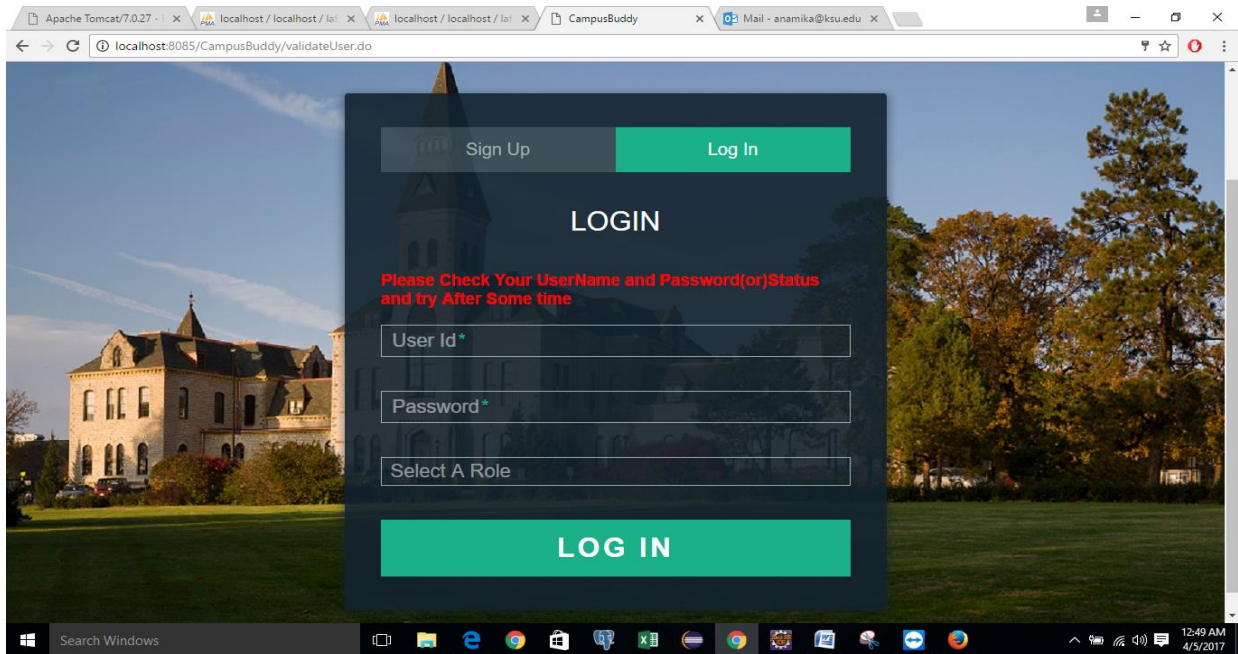


Figure 6.6 Trying to login without verification

As, the next process we send a mail to the student KSU email id, from a random email id which I created just for this purpose which some special conditions. The email is generally in the junk folder in the KSU email as university email has certain filters for external mails. The mail looks something like below, with a link. Once the link is clicked, the user is activated.

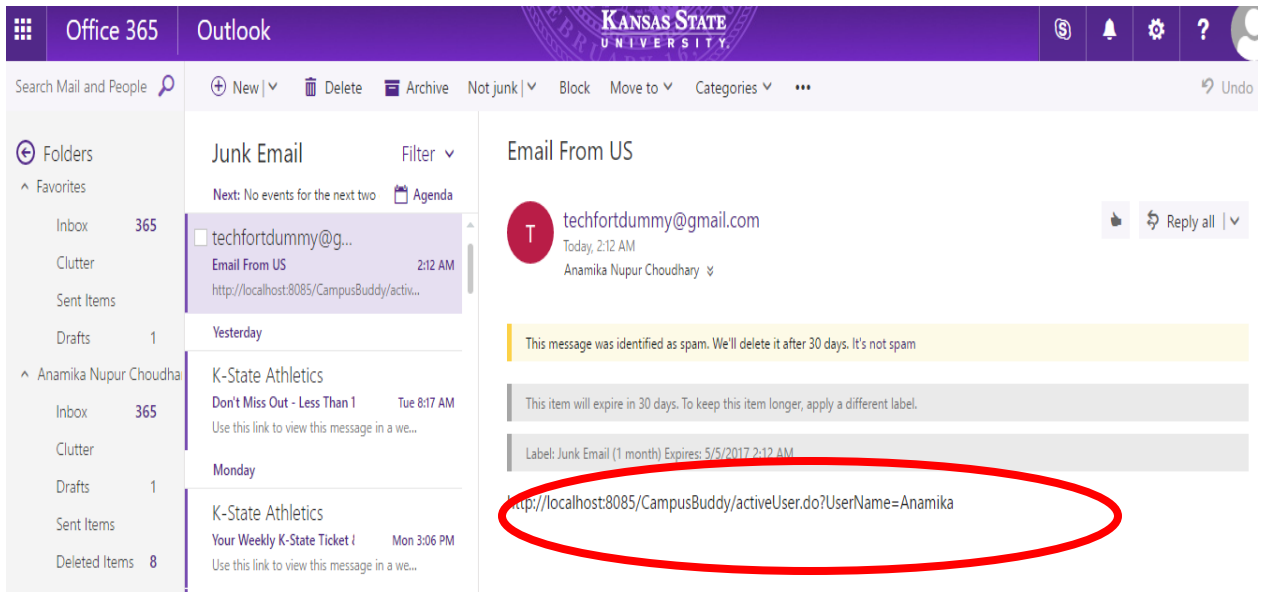


Figure 6.7 Verification code sent to KSU email id

At this point, the user is activated and the database will look something like figure 6.8 and it will give a message on the registration page as the user is activated, as in figure 6.9.

userName	emailid	password	role	phoneno	status
admin	admin@gmail.com	X+tGwEBkr7aJJSpQI9fUJg==	ADMIN	5655665552	Active
Anamika	anamika@ksu.edu	SHLwDw/NGYA=	STUDENT	7857706101	Active
Shravan	shravandg@ksu.edu	qr956UDejRs=	STUDENT	2486642345	Active
Prathap	prathap93g@gmail.com	b4WE4bXVBj0=	STUDENT	7306827578	Active

Figure 6.8 Status updated to “Active” in registration table

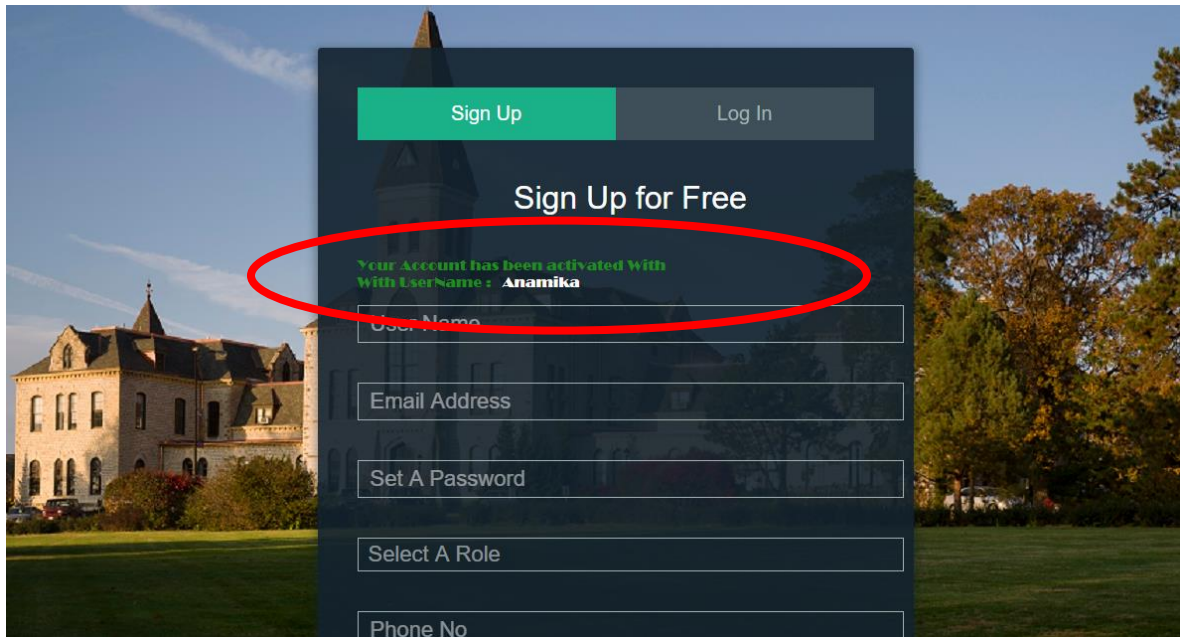


Figure 6.9 Message shown after verification of email

Once the student logs in with his login and password, the gets access to the application home page.

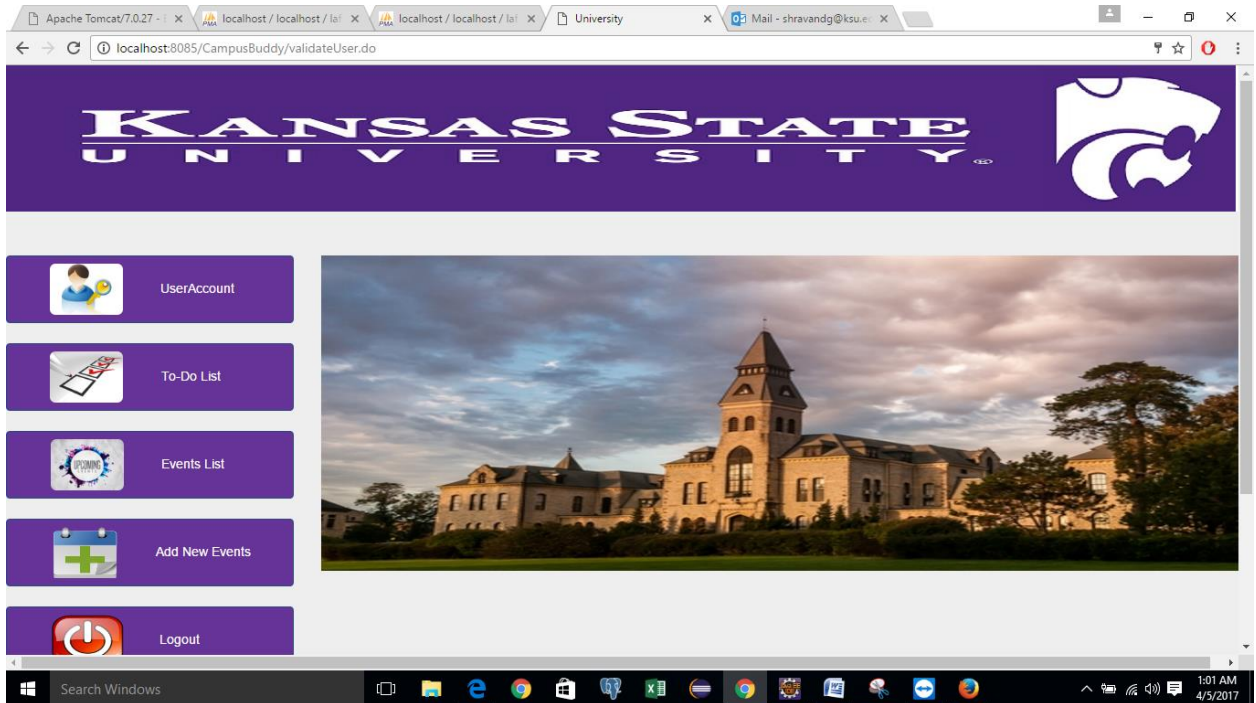


Figure 6.10 Students home page

The first tab **UserAccount**, gives the account details of the user/student. It is all the data of the student which stored in database are reflected.

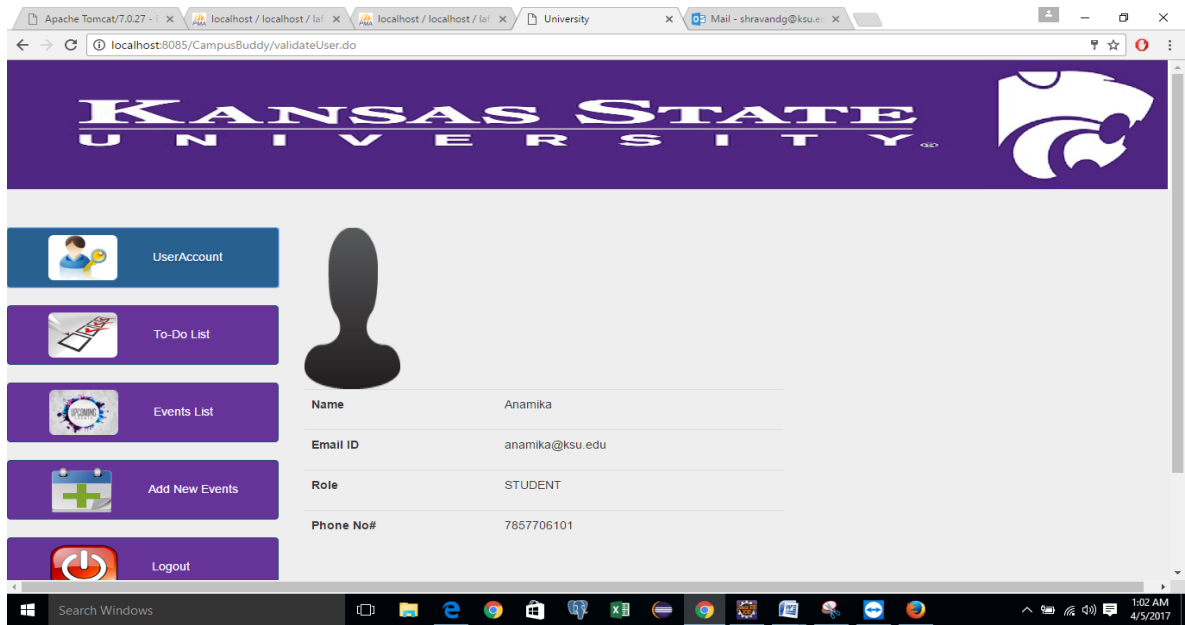


Figure 6.11 Student account info

	userName	emailid	password	role	phoneno	status
<input type="checkbox"/>	admin	admin@gmail.com	X+tGwEBkr7aJJSpQI9fUJg==	ADMIN	5655665552	Active
<input type="checkbox"/>	Anamika	anamika@ksu.edu	G9nczj4BwxlsxBJEP8pTBg==	STUDENT	7857706101	Active
<input type="checkbox"/>	sravan	shravandg@ksu.edu	b4WE4bXVBj0=	STUDENT	7306827578	Active
<input type="checkbox"/>	Prathap	prathap93g@gmail.com	b4WE4bXVBj0=	STUDENT	7306827578	Active

Figure 6.12 Student details which are stored in registration table

The second tab **To-Do List** will give the user the list of to-do's they should finish. It will in turn have the 2 separate blocks. Completed and To-Do's list. In the first tab there will be all the To-Do's student has already finished and the To-Do's tab will have to-do which they still have to finish.

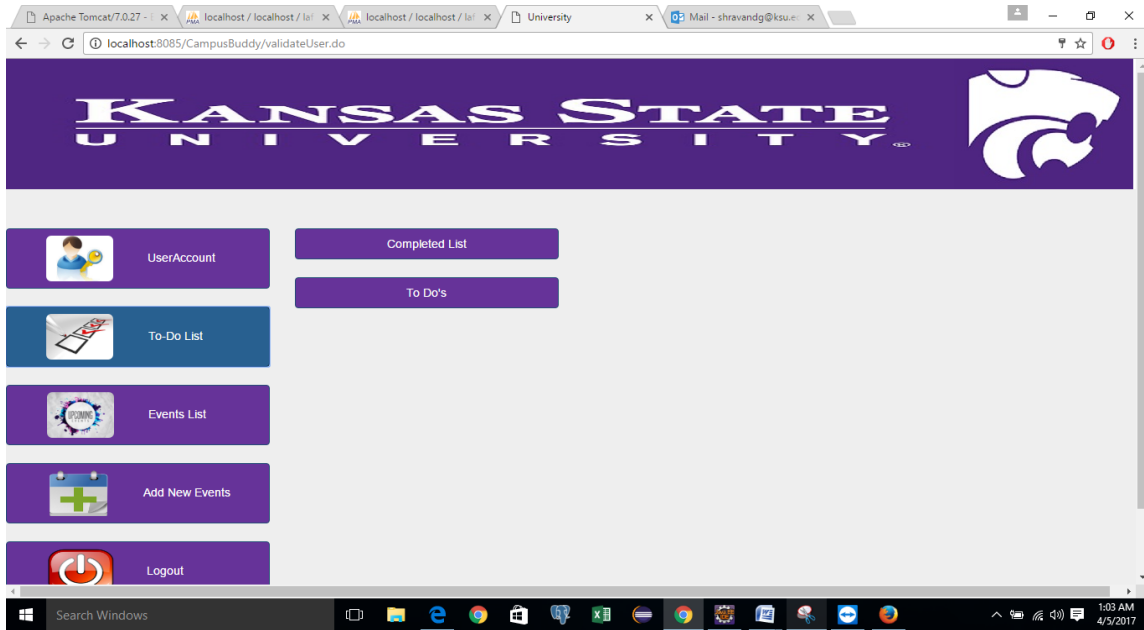


Figure 6.13 Student To-Do page

Once the student selects the To-Do's tab, all the to-do's added by the Admin are displayed. Each to-do is in a sequential order when selected displays all the details associated with it. It will only display the to-do which are not already completed by the student. When student selects To-Do tab all the other tabs are blocked, so that even if they are selected they do not perform any specific functionality.

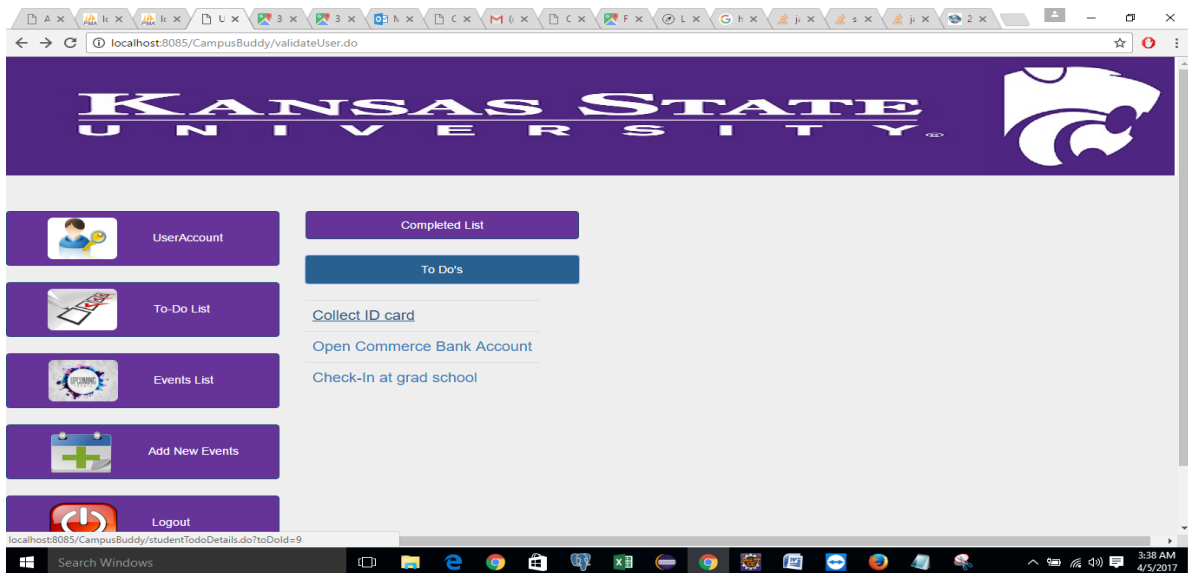


Figure 6.14 Student To-Do's not completed

In the above picture if we select any of the to-do, it will display the all the details and the map location as in figure. It will also have two checkboxes where we select completed or not completed.

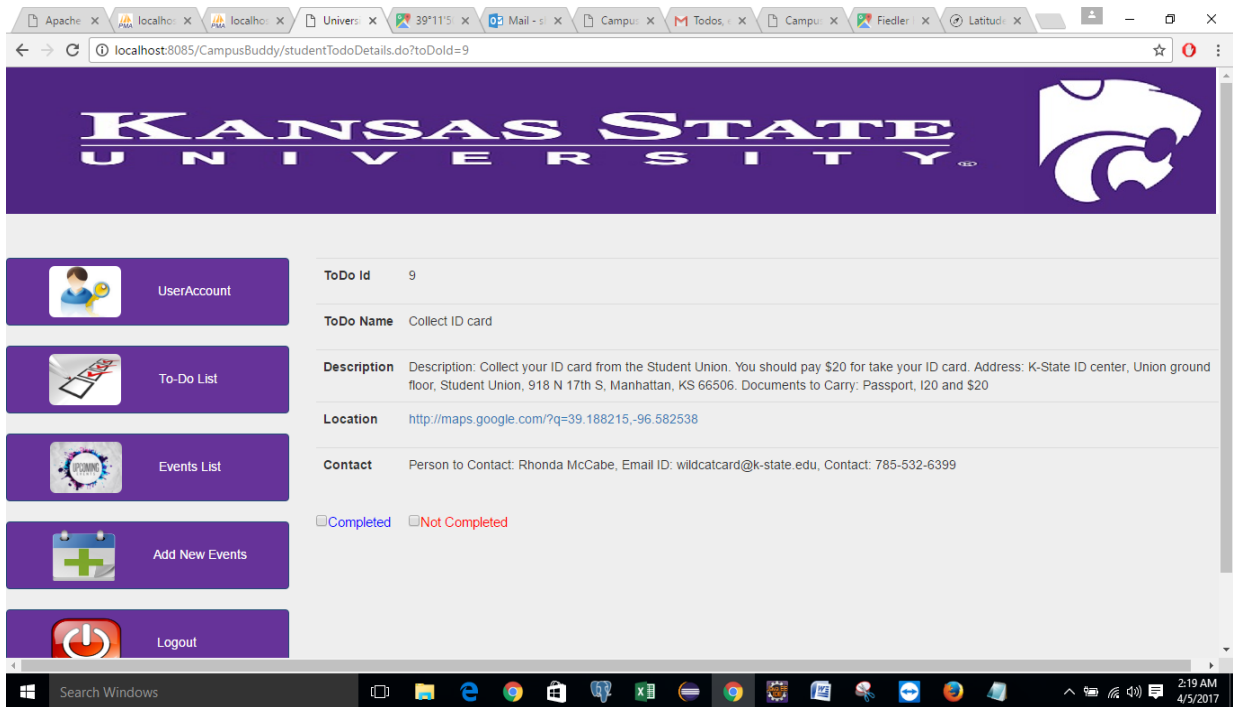


Figure 6.15 Student To-Do's details

The map location when selected open the google map location

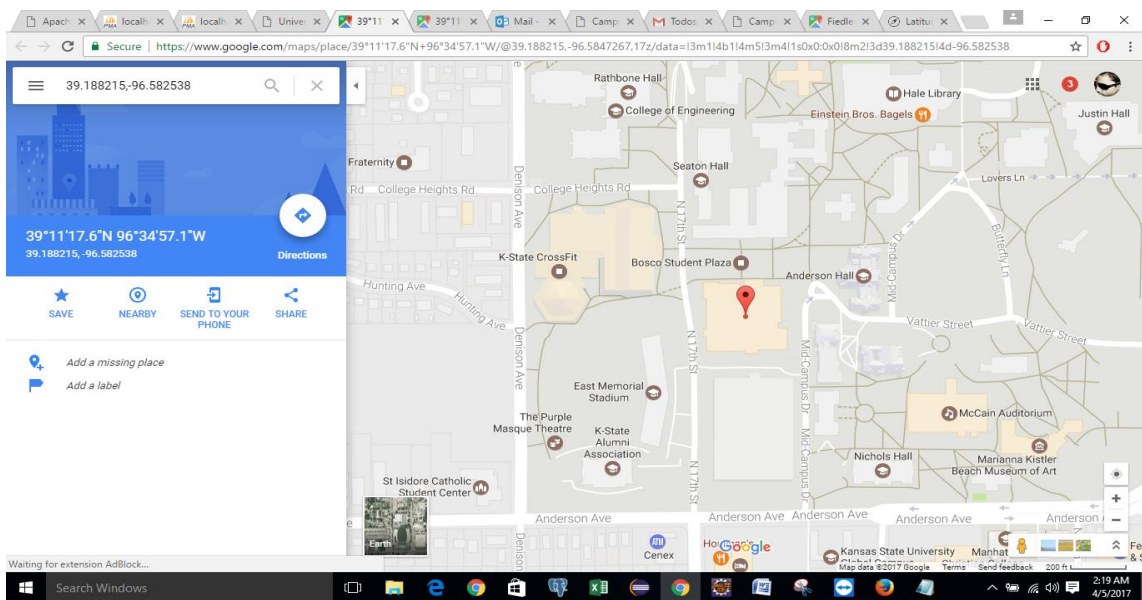


Figure 6.16 Map location

Once the student checks the completed tab, it moves to completed list and even the database is updated as completed as shown in 6.17.

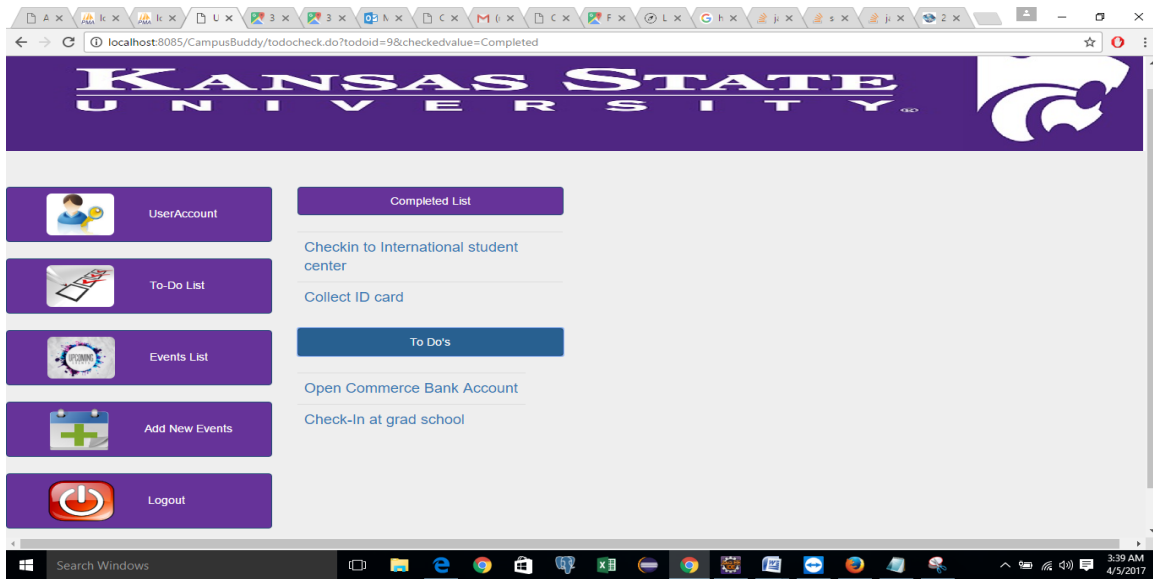


Figure 6.17 To-Do moved to Completed list

Once the to-do is moved to completed list we can no longer see the checkboxes. It is just there for reference.

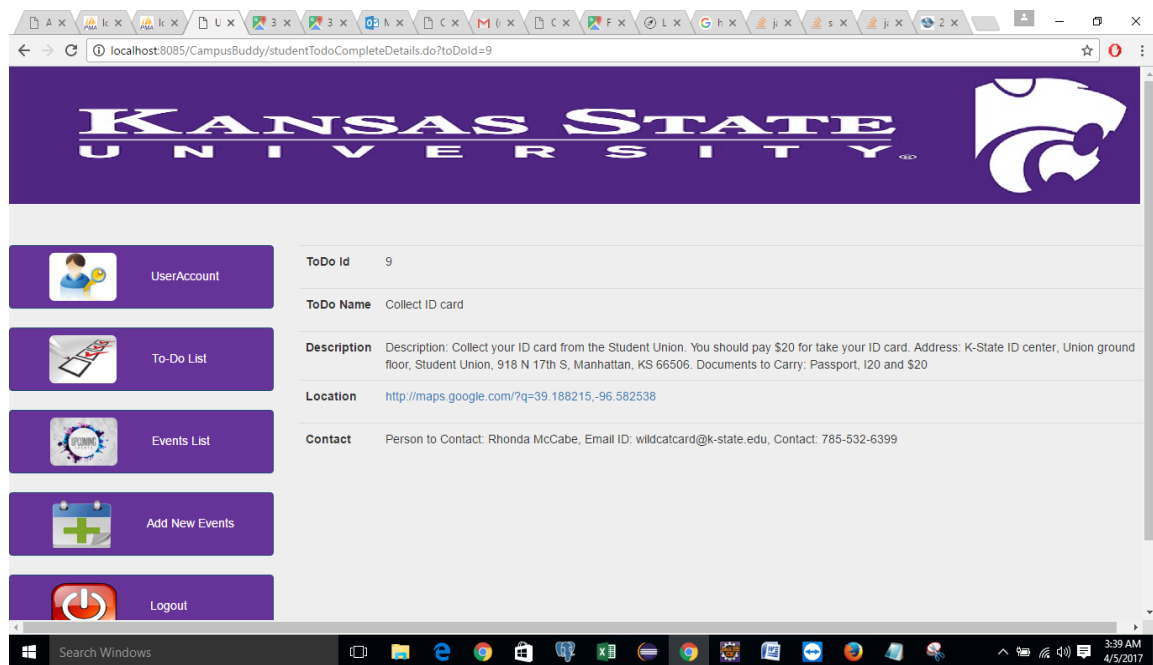


Figure 6.18 To-Do page with no option to select completed or not completed after moving to-do to the completed list

The event list when selected shows the events which are added by the admin. It is similar to the to-do list.

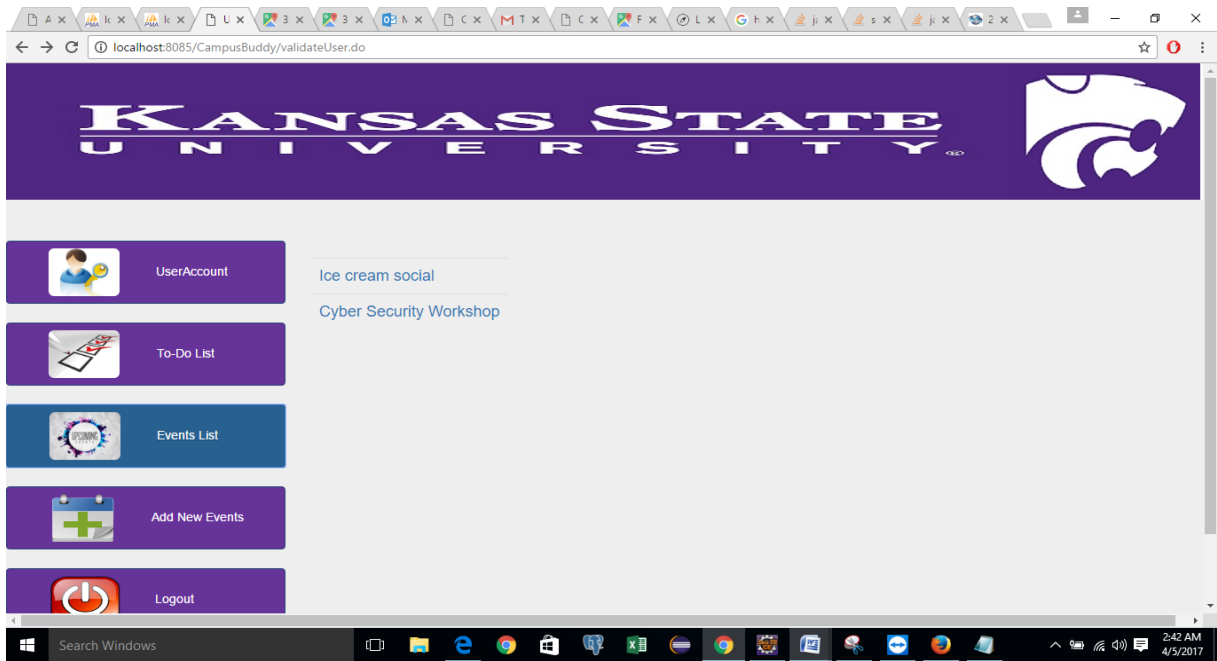


Figure 6.19 Events list

Each event when opened, will have 2 checked boxes which takes input from the student on if they are interested or not. Also, it will give the number of students interested in the event which will help the students take decision if they going for the event and they are not alone.

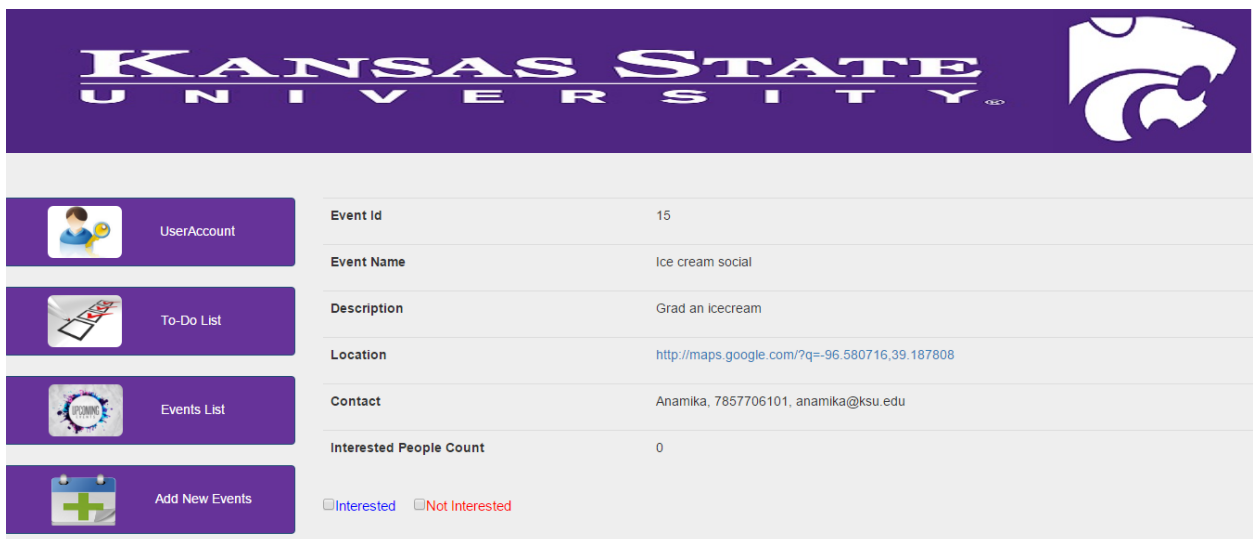
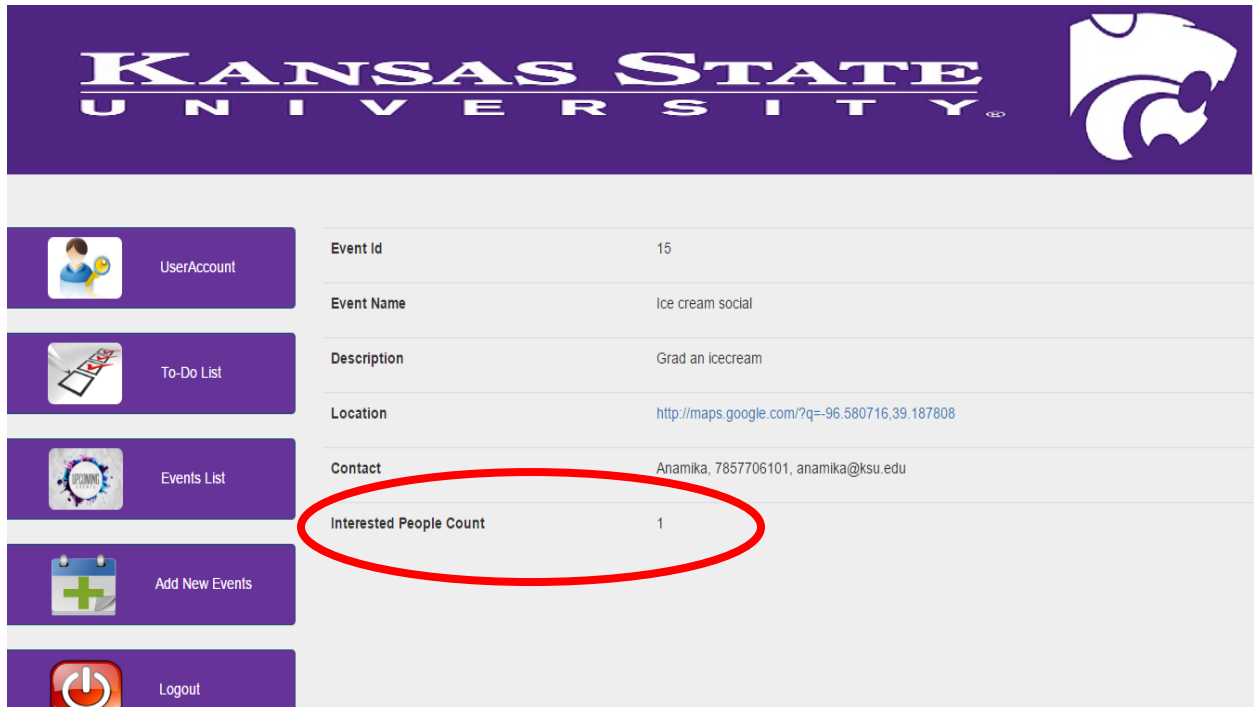


Figure 6.20 Event details

Now if the student selects interested, the interested people count will increase and the database **Eventsint** is updated.

	id	eventid	userName	status
<input type="checkbox"/>  	11	15	Anamika	Interested

Figure 6.21 Status after student selects interested for an event



The screenshot shows the Kansas State University website interface. On the left is a sidebar with navigation buttons: UserAccount, To-Do List, Events List, Add New Events, and Logout. The main content area displays event details for Event Id 15, titled 'Ice cream social'. The 'Interested People Count' is highlighted with a red circle and shows a value of 1. Other details include the description 'Grad an Icecream', a location link, and contact information for Anamika.

Field	Value
Event Id	15
Event Name	Ice cream social
Description	Grad an Icecream
Location	http://maps.google.com/?q=-96.580716,39.187808
Contact	Anamika, 7857706101, anamika@ksu.edu
Interested People Count	1

Figure 6.22 Interested people count increases

Students can also new events which they know of. They just have to enter all the details like latitude and longitude location of the place where the event is scheduled and save it. The event added by student is added in **sevents** table.

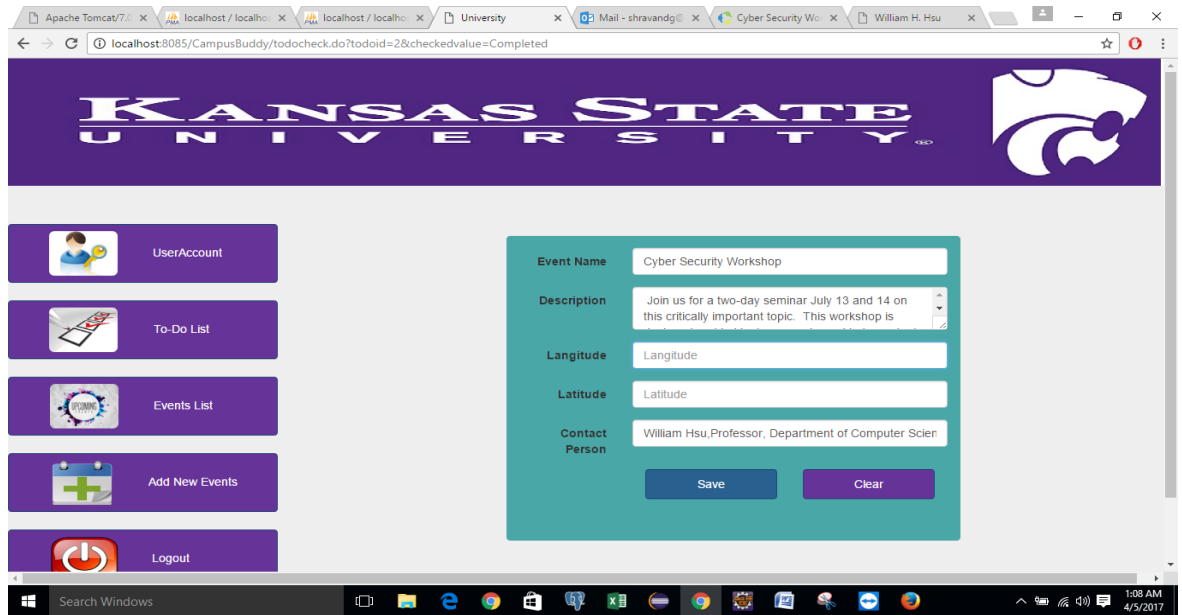


Figure 6.23 Student adds new events

studenteventId	studenteventName	studenteventdesc	studentlangitude	studentlatitude	studentcontactPerson
18	Ice cream social	Grad an icecream	39.187808	-96.580716	Anamika, 7857706101, anamika@ksu.edu

Figure 6.24 Event added in sevents table

6.1.2 Admin Module

The first index page is the same for both student and Admin. Then the admin logs into his account. There is not registration required for him. The first page of the admin is as below.

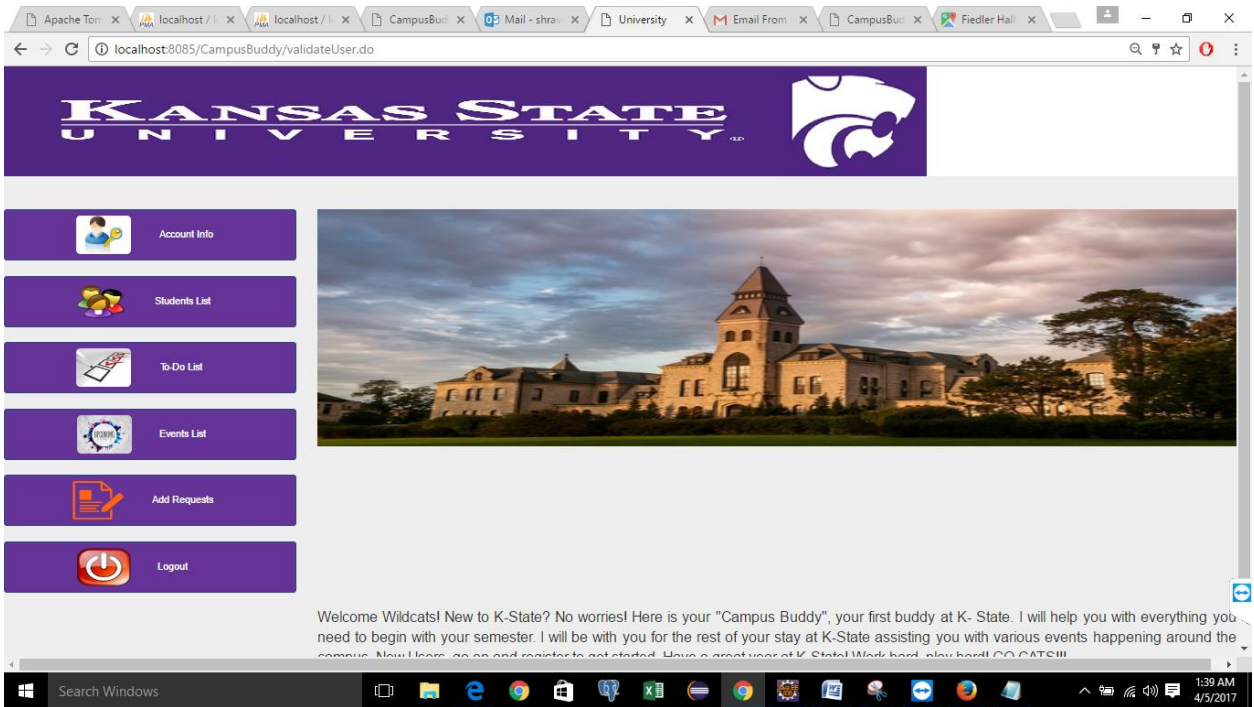


Figure 6.25 Admin home page

The admin can see the student list. It will show all the students who have registered, with all their details which they gave while registration.

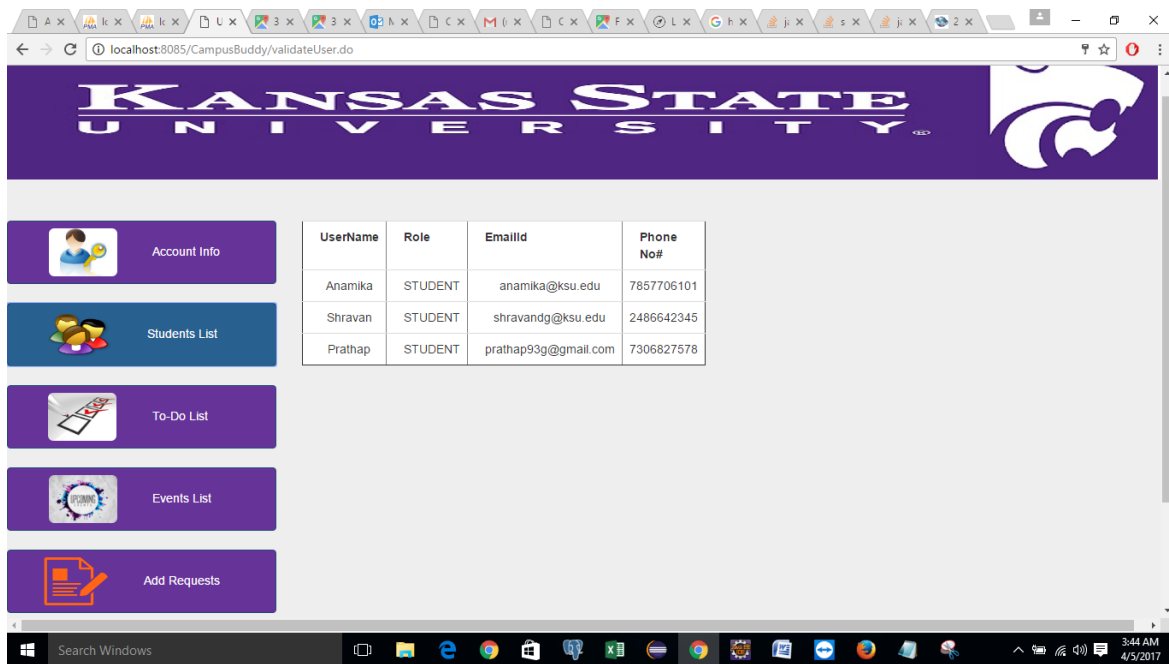


Figure 6.26 Students list

When the Admin selects To-Do List, he can see all the to-do's already added by him like in figure 6.26. He can also add a new to-do, update previous to-do list or delete them.

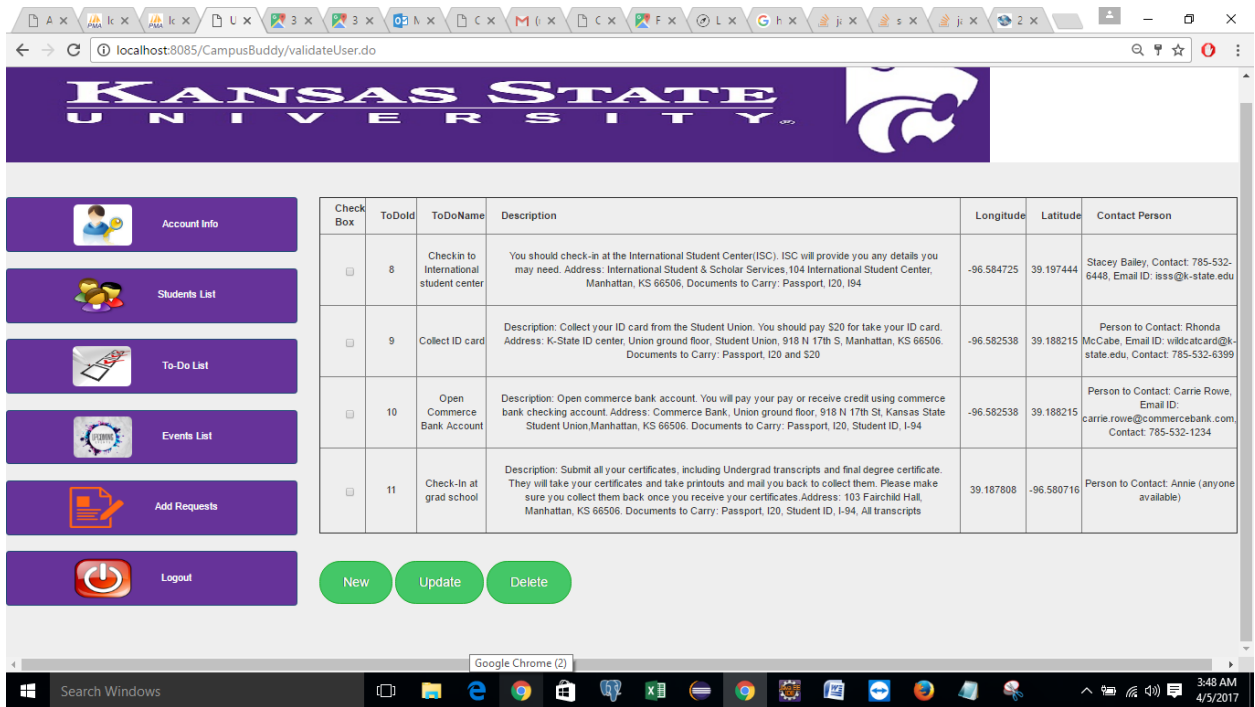


Figure 6.27 Admin side To-Do page

When the Admin selects New, he can enter new to-do as shown in picture 6.27 and save it.

While adding the Admin can add as much description as possible, with the location's latitude and longitude. He can also provide the contact information of the person responsible for handling the to-do. It is updated in todo table.

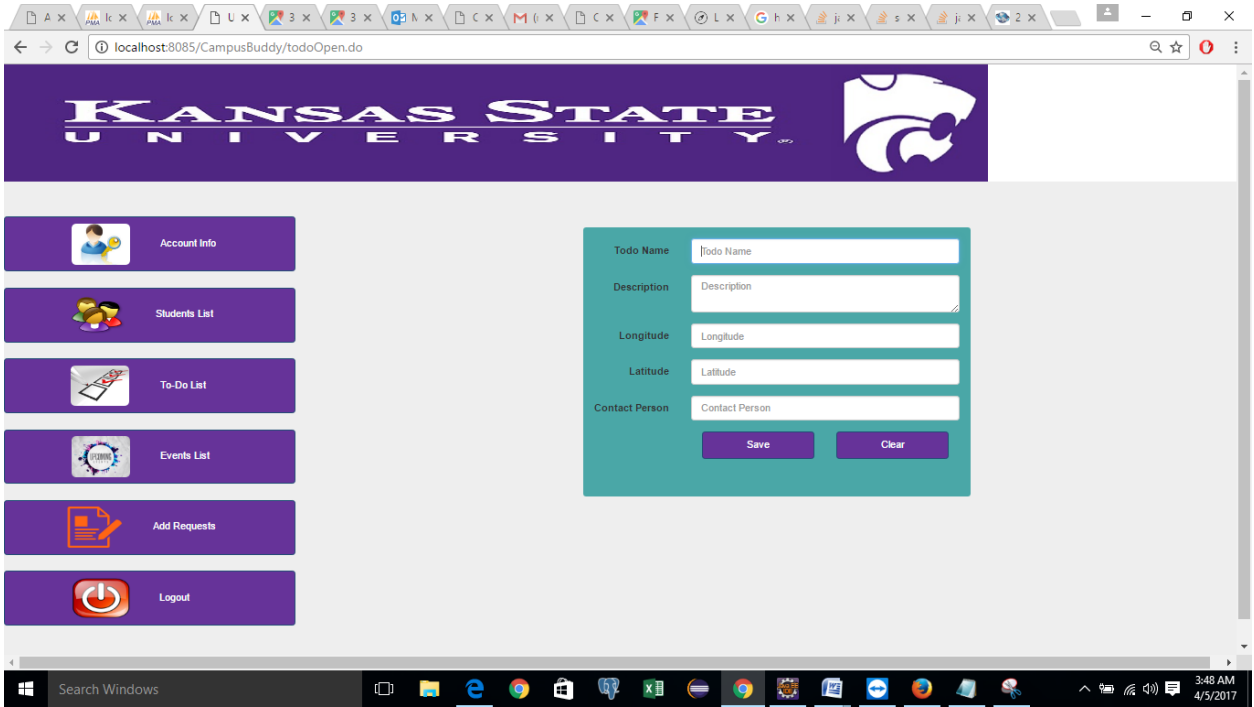


Figure 6.28 Admin adds new To-Do

todoId	toname	description	longitude	latitude	contact
8	Checkin to International student center	You should check-in at the International Student C...	-96.584725	39.197444	Stacey Bailey, Contact: 785-532-6448, Email ID: is...
9	Collect ID card	Description: Collect your ID card from the Student...	-96.582538	39.188215	Person to Contact: Rhonda McCabe, Email ID: wildca...
10	Open Commerce Bank Account	Description: Open commerce bank account. You will ...	-96.582538	39.188215	Person to Contact: Carrie Rowe, Email ID: carrie.r...
11	Check-In at grad school	Description: Submit all your certificates, includi...	39.187808	-96.580716	Person to Contact: Annie (anyone available)

Figure 6.29 To-Do added in todo table

After selecting the Events List, Admin can see previous events he added, update and delete them. He can add new events as well. It will give the Admin a new space to add them as in picture 6.30.

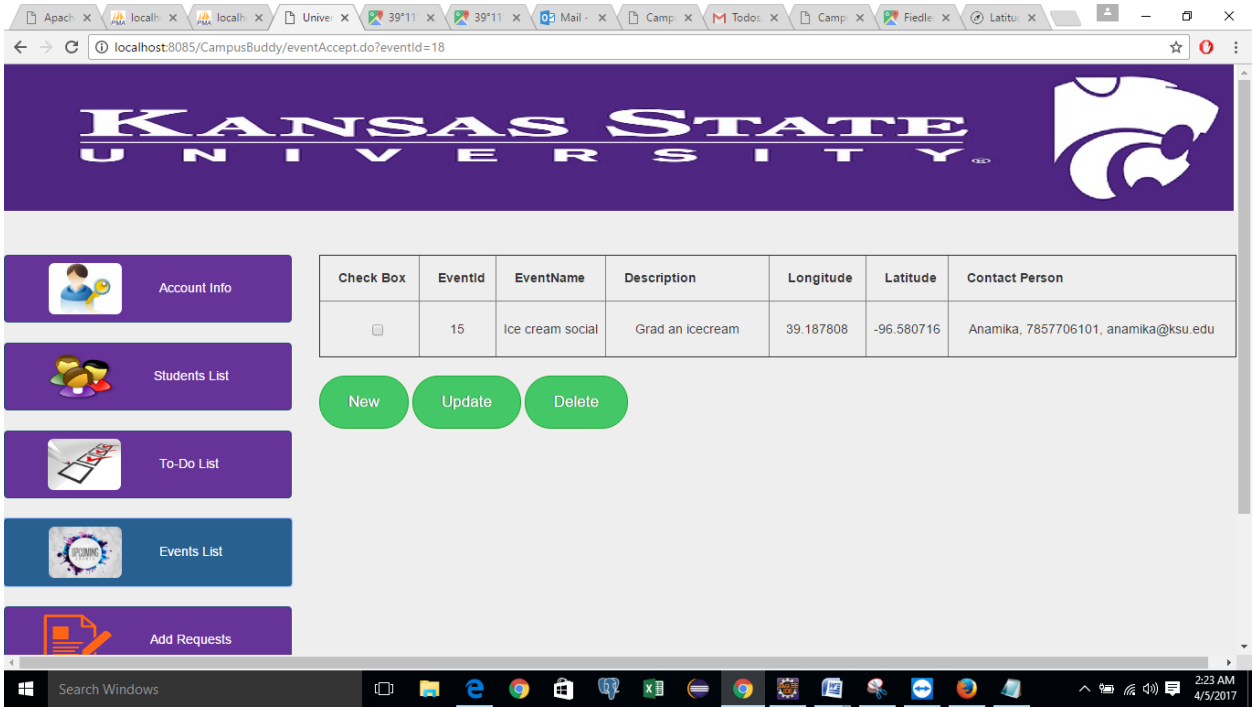


Figure 6.30 Admin events page

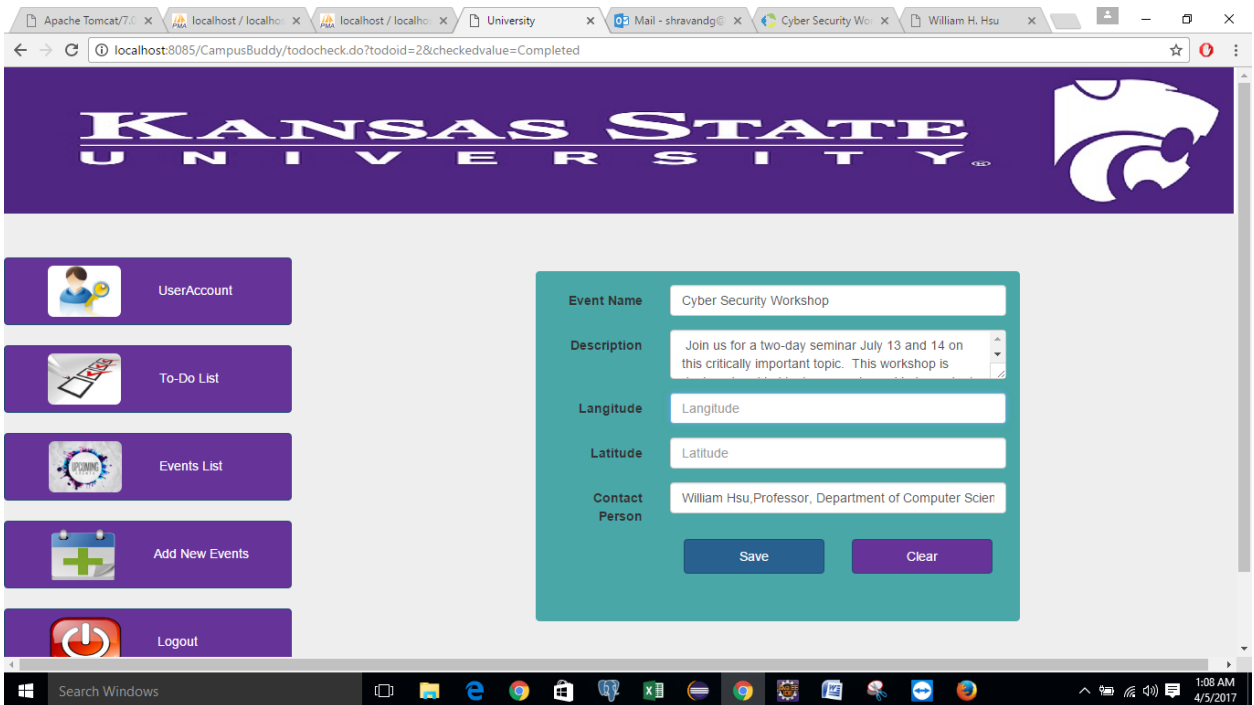


Figure 6.31 Admin adds new event

When a student adds an event, it is not directly added to the main events page, it is rather added to sevents page. Only after admin accepts it, the event is added to the events page. When the admin clicks **Add Requests**, he can see all the events student added and he can accept and reject it.

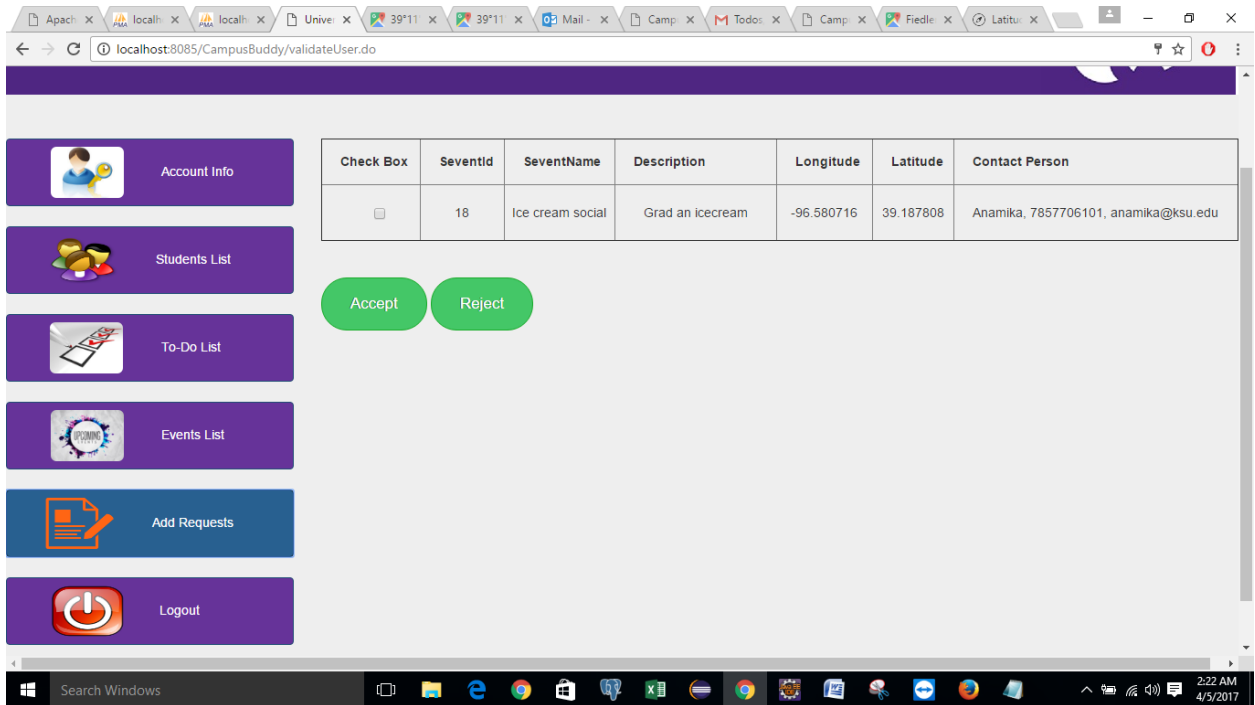


Figure 6.32 Admin accept or rejects student event add request

The logout just logs out the Admin from this page and redirects him to registration page.

Chapter 7 - Database Design

Designing a well laid database is as important as designing the frontend. A good database design is helpful in many ways. The main concept of designing a database is to arrange the required information into tables and these tables are then normalized.

Few advantages of having a good database design is

- ➔ Eliminating redundant data. E.g., we cannot have same user multiple times in the database. It also saves space.
- ➔ Database access becomes really as data is well arranged.
- ➔ Increased performance.
- ➔ Maintains data accuracy and integrity.
- ➔ More secure.

7.1 Database Tables

In this project, I have used 6 tables to store various details.

- 1) **registration** table: This table stores the details of user and admin which they insert while registration.

			userName	emailid	password	role	phoneno	status
<input type="checkbox"/>			admin	admin@gmail.com	X+tGwEBkr7aJJSpQI9fUJg==	ADMIN	5655665552	Active
<input type="checkbox"/>			Anamika	anamika@ksu.edu	G9nzcj4BwxlsxBJEP8pTBg==	STUDENT	7857706101	Active
<input type="checkbox"/>			sravan	shravandg@ksu.edu	b4WE4bXVBj0=	STUDENT	7306827578	Active
<input type="checkbox"/>			Prathap	prathap93g@gmail.com	b4WE4bXVBj0=	STUDENT	7306827578	Active

Figure 7.1 registration table

The password in this field is stored in an encrypted format. The main purpose of encryption here is to make sure that even if some external resource gets access of the database, he cannot see the actual password of the user. Also as any number of students can use this application, there can be

multiple student account. Also the verification link is sent to the mail id the student provided while registration. The data for the login page is also extracted from this page and matched.

- 2) **events** table: This table stores all the event details the admin enters for his end. It provides eventname, description and the location using latitude and longitude.

eventid	eventname	description	langitude	latitude	contact
15	Ice cream social	Grad an icecream	-96.580716	39.187808	Anamika, 7857706101, anamika@ksu.edu
16	Cyber Security Workshop	This workshop provides an opportunity to share ide...	-96.581025	39.188651	Amy Jackson, 7851234321, amy@ksu.edu

Figure 7.2 events table

- 3) **eventsint** table: This is the events interested table. It stores details of the students who are interested in a particular event. This table is joined to the events tables with the id. It gives information if the student is interested in any particular event or not.



	id	eventid	userName	status
<input type="checkbox"/>  	11	15	Anamika	Interested

Figure 7.3 eventsint table

- 4) **sevents** table: This is the student's events table. All the events student inserts from his home page are added to this table. Once the admin approves this request the event is added to the main events table.

studenteventid	studenteventName	studenteventdesc	studentlangitude	studentlatitude	studentcontactPerson
18	Ice cream social	Grad an icecream	39.187808	-96.580716	Anamika, 7857706101, anamika@ksu.edu

Figure 7.4 sevents table

- 5) **todo** table: This table has the to-do list stored with the description, longitude, latitude and contact information. This basically has all the to-do's which are and which are not completed by the students.

todoId	toname	description	longitude	latitude	contact
8	Checkin to International student center	You should check-in at the International Student C...	-96.584725	39.197444	Stacey Bailey, Contact: 785-532-6448, Email ID: is...
9	Collect ID card	Description: Collect your ID card from the Student...	-96.582538	39.188215	Person to Contact: Rhonda McCabe, Email ID: wildca...
10	Open Commerce Bank Account	Description: Open commerce bank account. You will ...	-96.582538	39.188215	Person to Contact: Carrie Rowe, Email ID: carrie.r...
11	Check-In at grad school	Description: Submit all your certificates, includi...	39.187808	-96.580716	Person to Contact: Annie (anyone available)

Figure 7.5 todo table

6) **todocomp** table: This table is joined to todo table by the todo id and username from registration table. It has information about which student has finished which to-do. All the to-do's completed and the to-do pending are shown as two separate tables to the user.

			userName	todoId	status
<input type="checkbox"/>			Prathap	2	Not Completed
<input type="checkbox"/>			Anamika	2	Completed
<input type="checkbox"/>			Anamika	8	Completed

Figure 7.6 todocomp table

7.2 ER Diagram

Entity Relation diagram, gives the relation between various entities, in short relationship between various tables in the database. It shows that how every tables in maintained in the database and how are related to each other.

In this project, registration, events and todo tables are independent. But eventsint table has eventId as well as student name associated with the status to the event he added (i.e., whether it is interested or not). Todocomp table has student name from registration table and todo id from todo table associated with it along with status (i.e., whether it is completed or not). sevents table has the details of the events provided by the students.

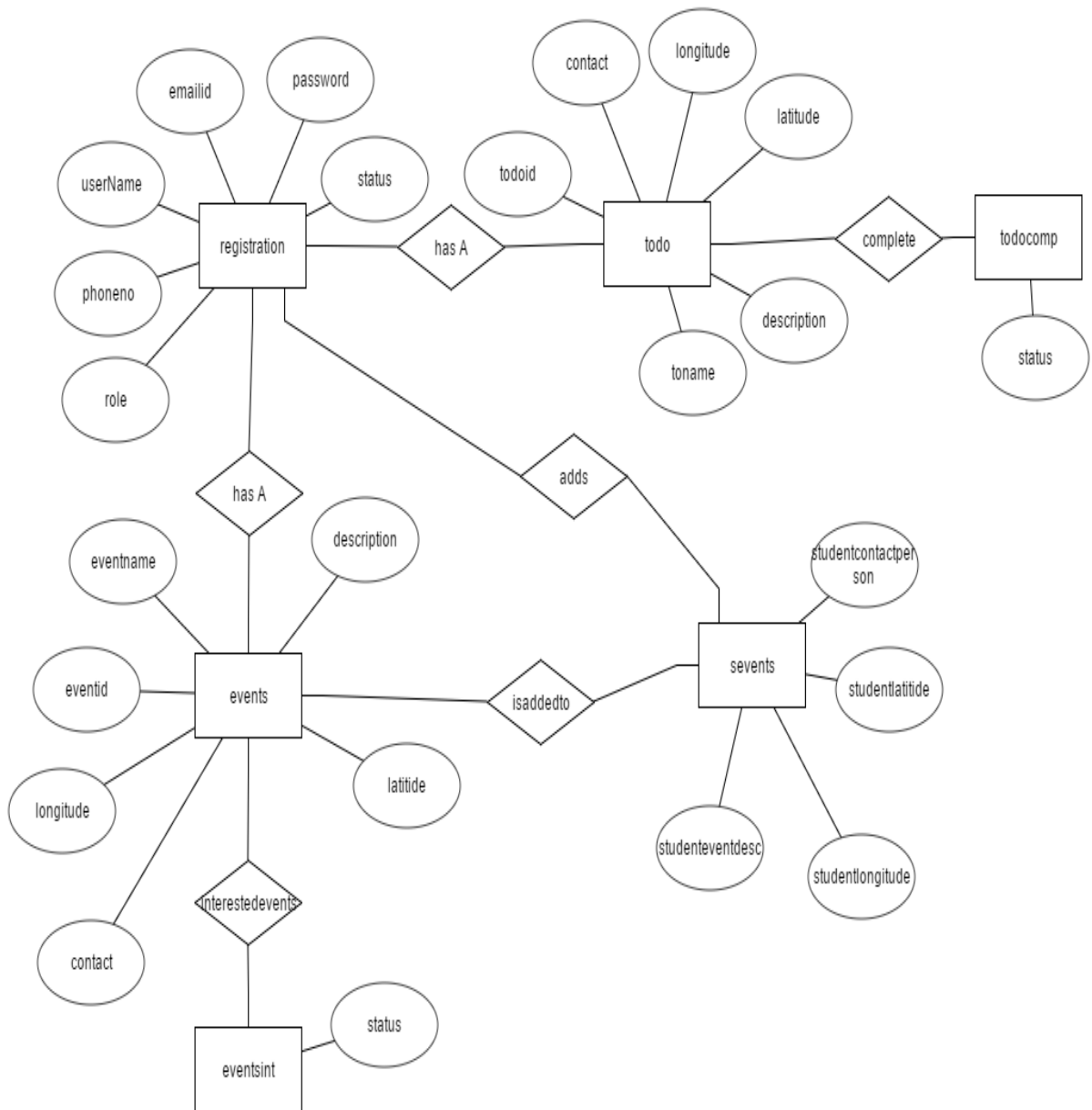


Figure 7.7 ER diagram

Chapter 8 - Testing

Code is written and developed by a human. Anything which is man-made is prone to errors. Not all errors are harmful for your application, but, few are. So we should make sure that the application has no scope for errors. To make sure the app is clean, we need to test our code while and after development. Testing makes sure that the application is clean. There are many testing levels available at different levels.

8.1 Testing Levels

Unit Testing: It is testing individual units of the code. This testing is usually done at developer end and it done while developing the code. At this stage, testing is done at individual code level, in isolation to other parts of the program. In this project at each level unit testing is performed to make sure they work as expected.

Integration Testing: After unit testing, we perform integration testing where we try integrate individual units together. There are situations when individual units may work as expected but when we combine these units they may fail. Hence, we should make sure integrated units works fine and produce the desired result.

System Testing: After all the integration testing is done, we check if the whole integrated system meets all the requirements.

Regression Testing: Regression testing helps developer to find the problems which he encounters after he fixes certain bugs, or by changing the environment. Hence every time we change anything in the code, we should make sure we check the whole system again.

Acceptance Testing: This testing is conducted to check if the output/result obtained from the code is as expected and match the business requirements. In short, if it should be accepted.

Load Testing: Load testing is the process in which we make the system work to its maximum capacity and check if it can handle the load. It helps us full capacity of our code.

Performance Testing: Performance testing is the process where we check the performance of the system under a particular workload and check for the performance of the system.

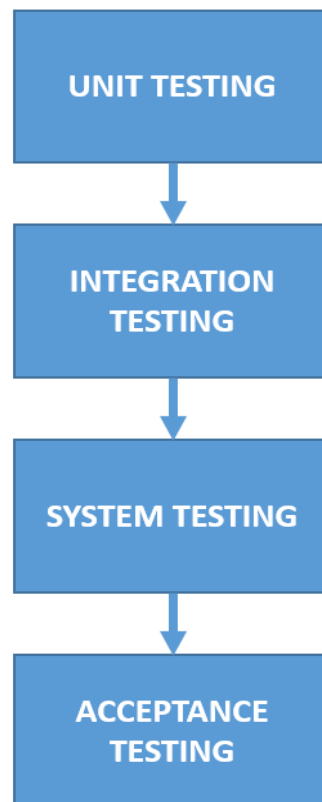


Figure 8.1 Testing Levels

8.2 Tests performed on student/user module

MODULE	TEST CASE	EXPECTED RESULT	TEST RESULT
User/Student	Provide any other email id except K-State email id with @ksu.edu domain	Throws error and registration not successful	PASS

User/Student	Provide valid KSU email id with @ksu.edu domain	User successfully registered message and is stored in registration table but with status inactive .	PASS
User/Student	User tries to login before activating or verifying the email.	Throws error and is not allowed to login	PASS
User/Student	After registration student gets mail for verifying his account	Email received to student KSU email id	PASS
User/Student	Student clicks the link provided for verification	It will activate user and on the registration page, it shows student activated.	PASS
User/Student	Student login after account verification.	Student is allowed to login after he verifies his account	PASS
User/Student	Student enters wrong username and password	User login unsuccessful	PASS
User/Student	Student enters correct username and password but enters the role as Admin	User login unsuccessful	PASS
User/Student	Student redirected to student homepage after login	User can view his home page	PASS
User/Student	Student clicks on UserAccount button	User can see all the details he/she entered while registration.	PASS

User/Student	New student clicks on Completed List	It should be empty.	PASS
User/Student	Student clicks on To-Do button	User can see two tabs Completed List and To-Do's.	PASS
User/Student	New student clicks on To-Do's button	Student is able to see the To-Do list	PASS
User/Student	Student clicks on one of the To-Do's	Displays all the information related to the To-Do	PASS
User/Student	Student clicks the map link provided for the To-Do.	Takes the user to google maps page.	PASS
User/Student	Every To-Do in the To-Do's tab is provided completed or not completed checkbox	Displays the completed or not completed checkbox along with To-Do description.	PASS
User/Student	Students selects completed checkbox.	To-Do moved to completed list and also updated in the database	PASS
User/Student	Returning student clicks on Completed List button	Displays the To-Do marked as completed by the user previously.	PASS
User/Student	Student clicks on Events List button	Displays all events in DB.	PASS

User/Student	Student clicks on any event.	Displays the details related to the event along with interested and not interested checkbox.	PASS
User/Student	Student selects on interested/not interested checkbox.	Updates database accordingly.	PASS
User/Student	Student clicks Add Events button	Provides user a form to enter all the events details along with latitude and longitude.	PASS
User/Student	Student clicks Add Events button	Provides user a form to enter all the events details along with latitude and longitude.	PASS
User/Student	Student clicks clear button before saving on the vents page.	Event not saved in the sevents database	PASS
User/Student	Students hits save after entering events details.	Student event saved in sevents table, but not updated in main events table.	PASS
User/Student	Student clicks Logout	User is logged out of the app.	PASS

Table 8-1 Testing on Student Module

8.3 Tests performed on Admin module

MODULE	TEST CASE	EXPECTED RESULT	TEST RESULT
Admin	Admin provides wrong username and password for login	Admin login unsuccessful and throws an error message	PASS
Admin	Admin enters correct username and password but enters the role as Student	Admin login unsuccessful	PASS
Admin	Admin enters correct username and password	Admin login successful	PASS
Admin	Admin redirected to admin homepage after login	Admin can view his home page	PASS
Admin	Admin clicks on AccountDeatils button	Admin can see all his details.	PASS
Admin	Admin clicks on StudentsList button	Displayed students list	PASS
Admin	Admin clicks on To-Do List button	Admin is redirected to To-Do page and displays all the To-Do previously added(in any)	PASS
			PASS

Admin	Admin clicks on Add button in the To-Do page	Open a new To-Do form for Admin to add details	
Admin	Admin clicks save button on the form	To-Do is added to todo table	PASS
Admin	Admin selects previous added To-Do check box and click update	Admin can edit details of the To-Do selected.	PASS
Admin	Admin selects previous added To-Do check box and click delete	To-Do is deleted.	PASS
Admin	Admin clicks on Event List button	Admin is redirected to Event page and displays all the events previously added(in any)	PASS
Admin	Admin clicks on Add button in the event page	Open a new form to enter event details	PASS
Admin	Admin clicks save button on the form	Event is added to events table	PASS
Admin	Admin selects previous events check box on the events page and click update	Admin can edit details of the Event selected.	PASS
Admin	Admin selects previous events check box on the events page and click delete	Event is deleted.	PASS
Admin	Admin clicks Add Events Button	Displays all events added by the students	PASS

Admin	Admin selects any particular event and clicks Accept	Event is added to events table	PASS
Admin	Admin selects any particular event and clicks Reject	Event is rejected.	PASS
Admin	Admin clicks Logout	Admin is logged out of the app.	PASS

Table 8-2 Testing on Admin Module

8.4 Performance Testing

Performance testing is performed to determine the speed or effectiveness of an application. It involves many quantitative tests for calculating response time or number of millions of instructions per second at which the system functions. This testing is useful because when we deliver the final product, we should make sure the app is available to any number of users and does not just stop working when there is heavy load. Hence, we need to make sure it does function for different sets of users. To perform this testing, I used JMeter, which is one of the tools available to check the performance of web applications. As, this application is running on local host and can just see the performance of database and how it is able to handle multiple users. The table 8.3 shows the throughput and average response time for a set of users and loop count.

	Number of Users(Threads)	Ramp-Up Period	Loop Count	Avg Response Time	Throughput
Observation 1	500	10	1000	2.47 sec	2918 res/min
Observation 2	1000	10	1000	2.40 sec	5736 res/min

Observation 3	1500	10	1000	2.25 sec	8314 res/min
----------------------	------	----	------	----------	--------------

Table 8-3 Performance Observation

Chapter 9 - Security

Security is the process of securing app from all the threats. Security is really important for any application, but most importantly in web applications as they run on external networks. If the data is not secure, it can pose serious security threats to your application. With so any options available online, hacking into any application is really simple, and can expose all your private information to the hacker. He can use your confidential information and play around with the database. For a secure application, we need follow 4 important principles. Confidentiality, Integrity, Availability and Nonrepudiation. In this app (Campus Buddy), hacker can get Admin details and update the to-do and event location to some fake location and confuse other students. To make sure these things never happen, I have provided basic security features to my app.

Below are some basic validation applied to this code.

Password encryption in the database (Confidentiality)

All the passwords in the database are encrypted. We use DES algorithm for encryption. It main purpose of password encryption is that people who have access to the database cannot see user password and misuse it. So it is always advised to encrypt the password in the database.

Apart from these there can be other security features which are useful for any web application

Email verification

To make sure we do not receive any malicious data from random users, this app is restricted to K-State students with valid K-State email id (@ksu.edu). Also, student should verify his email through the verification link which is sent to his KSU email id.

A separate user and Admin (Integrity)

To make sure no data is modified, we just have one admin who is capable of access, deleting, adding and updating information. This makes sure data is not changed by any other user.

Providing secure password for each user (Availability and Nonrepudiation)

To provide independent access to each student, every student should have a unique username and password, which he can use for while logging into this app. It makes the app available for a genuine user.

Apart from these security features there are other security threats which you should make sure is handled by your system.

9.1 Security issues in web application and their handling methods

SQL Injection

It allows a hacker to get crucial data from the server's database. SQL Injection can get the intruder to the application page without the need of actual id and password. He just inserts a query which is always true in place of actual username and password fields and it gives him access to admin home page, where he can modify data.

Example query: "SELECT * FROM tablename WHERE username = '' or '1=1'";

The above query is always true.

Countermeasures for the above problem:

- ➔ By making sure, user has not been provided all the privileges and has only limited permissions.
- ➔ By using PreparedStatement in the database connection. It will prevent injection attacks.

Username Enumeration

This kind of attack happens, when the entered input is wrong and the intruder keeps guessing the input unless he gets the actual one. It happens when admin or the user use basic, simple and common username and password like admin/admin or repetitive numbers like 123456 as password.

Countermeasures for the above problem:

- ➔ Creating strong password.
- ➔ Not allowing same username and password.
- ➔ Not allowing user name as password.
- ➔ Not allowing continuous numbers like 123 or repetitive numbers like 222 in the password.

Redirecting to fishy page and tampering parameters

In this attack the intruder changes the url link internally and redirect the user to a fishy page where user enters the information and it provides the whole information of the user to the intruder.

Countermeasures for the above problem:

- ➔ Making sure that the page is secured. (https)
- ➔ Using firewall.

Chapter 10 - Conclusion

In conclusion, I want to mention that, though K-State has various applications available to help students with every day needs, this system would be immensely helpful for any student who is new to K-State. The student will have all the details in one place, without having to collect the pamphlets or to keep track of the emails. They will never be misguided by anyone and will never miss any event or a deadline. As they are new to the place, they will also get the location map to the events and places where they need to go to finish their mandatory to-do.

This project also avoids malicious updates by notorious students by giving all the update permissions to the admin. Even if the student provides new event details, it should be accepted by the admin before being added to the event list. This application also restricted to K-State students, with a valid K-State mail id, with extension “@ksu.edu” to provide security from external intruders.

Also, all thanks to this project, I have a very strong understanding of web applications. I developed the whole app end-to-end, using AngularJS (used for verification and validation), HTML, CSS, JQuery (for the front-end), Spring Framework for all the development and MySQL as the back-end database, and all these technologies are highly preferred for web development. This project also provided me enough scope to research about the current trends in IT development, boosting my confidence and providing enough understandability of leading trends in web development.

Chapter 11 - Future Work

In this project there is lot of scope for future enhancement. The main purpose of this project was to help students during their initial days in the school. This app probably provides all the functionalities a student will need to complete the mandatory set of requirements before they enroll and start their classes, but there are several ways we can enhance this application. We can implement this project as a hybrid app, which will run on any kind of platform, just not the web. My intention to make this app run specifically on web was to learn web technologies (Springs framework in particular).

As part of my implement project I had similar setup using Android. We can use web services and integrate both Android and Web app. This integration will replicate everything we do on Android app of the web app and vice versa. We can add and view details from both Android and web platform.

Currently, this app is restricted to only events and to-dos' list for the students. But there are multiple other ways this app can help them. We can add details about workshops, Club with their locations and POC. We can integrate student syllabus, class timings, exam dates, final project submission dates all into this app. The UI can we further enhanced to make it look more user friendly. We can add comments section to the student's module, which will make this app more dynamic. Like students can update the details of the events, if the location provided is wrong, contact information is improper and many others. This would help other students and the admin to get the updated information about the app.

Likewise, the project can be improved in various ways to provide quality information to the students.

Bibliography

- [1] Vardi, E. (n.d.). *SlideShare*. Retrieved on 04/03/2017 from LinkedIn:
<https://www.slideshare.net/EyalV/angularjs-architecture>
- [2] Smith, J. (n.d.). Retrieved on 04/03/2017 from <http://www.informit.com>:
<http://www.informit.com/articles/article.aspx?p=2252414>
- [3] (n.d.). Retrieved on 04/03/2017 from blog.angular-university.io: <http://blog.angular-university.io/introduction-to-angularjs-form-validation/>
- [4] Bernado. (n.d.). *wijmo.com*. Retrieved on 04/03/2017 from <http://wijmo.com/blog/easy-form-validation-in-angularjs/>
- [5] Ruebelke, L. (n.d.). Retrieved on 04/03/2017 from Envato.com:
<https://code.tutsplus.com/tutorials/5-awesome-angularjs-features--net-25651>
- [6] (n.d.). Retrieved on 04/03/2017 from www.Pluralsight.com:
<https://www.pluralsight.com/blog/software-development/tutorial-angularjs-mvc-implementation>
- [7] (n.d.). Retrieved on 04/03/2017 from
<http://stackoverflow.com/questions/13067607/angularjs-client-mvc-pattern>
- [8] (n.d.). Retrieved on 04/04/2017 from www.javatpoint.com:
<http://www.javatpoint.com/spring-modules>
- [9] (n.d.). Retrieved on 04/04/2017 from <https://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#overview-getting-started-with-spring>
- [10] Rajput, D. (n.d.). Retrieved on on 04/04/2017 from www.dineshonjava.com:
<http://www.dineshonjava.com/2012/12/spring-web-mvc-framework-chapter-38.html>