# Software Design Specification Document (CS360)

# NOTICEBoard

**Group Number:** 4

**Adnan Ahmed**
**Anusha Cassum**
**Shayan Ali**
**Umar Farooq**
**Zain Imran**

**Course:** Software Engineering CS360

**Instructor:** Suleman Shahid

**University:** Lahore University of Management Sciences (LUMS)

**Version: 1.0**
**Date: (02/03/2017)**
**Number of hours spent on this document:** 60

# Contents

# 1 Change Log

## 1.1 Project Scope

In this project our goal is to create a unified platform for campus related notifications for academic communities, as opposed to the numerous other platforms currently under use. Outlook's Students' Events' uses a generic email to send all sorts of unsolicited information and is not a favorable platform for keeping track of events. Emails are not classified into categories. Facebook, another common medium for student activities also lacks organization and is often hard to navigate. For the case of the LUMS community, 'The LUMS Discussion Forum' on Facebook is always flooded with posts about seminars, trips, course and book swaps and it often takes some time to find the desired item despite the search option.

We introduce the concept of a "Notice Board" that will allow students to communicate among each other more effectively. The product will organise useful and relevant information scattered across all the above mentioned platforms into a single, unified platform, thus providing ready access to more organized information about everything campus related. The idea is inspired by a real noticeboard for schools that holds important information about the activities taking place in the institution. Each student will have access to the noticeboard and can use it to communicate to the rest of the LUMS community. Furthermore, the product will be free and restricted to the use for the LUMS community therefore providing a more dedicated environment for users. It will also minimize time wastage and improve communication efficiency.

## 1.2 Change log

After re-analysing the SRS and starting initial development, we found that most of the security requirements that were mentioned earlier in non-functional requirements are out of the scope of this project. This was decided, since our primary purpose in this project is user convenience rather than providing a platform which is highly secure. It is therefore decided to leave out the following requirements:

- Notification of multiple failed account authentication attempts so as to warn the user in case if someone is trying to guess the password.
- Prevention of any malware being downloaded to the user's host machine via the server. This may be done by a firewall at the server side but is still subject to convenience in the development phase.
- Deactivation of account in case if a user is graduating and/or is no longer concerned with the activity on the platform in order to prevent loss of integrity and identity misuse.
- Shifting from Google Calendar to a custom built calendar since Google Calendar requires login to Gmail. People reluctant to trust with their gmail credentials on a third party entity. Purpose of noticeboard not to integrate google services, but to organize campus related events. So integration with gmail out of scope. Also not everyone uses GMail or Google Calendar.

# 2 Introduction

## 2.1 Document Purpose

Project release #: 1.0.0
Our project, aptly named “NoticeBoard” will allow users to communicate with each other effectively by allowing easy access to updates, events and notifications, all from a single source, saving precious time and energy. This would be in lieu of the cluttering users have to face in their email boxes and would present all important information in a more organized manner.

This document will formally highlight the software overview and constraints of the project, the system architecture, Data Structures used and all Database Models with detailed design that will be followed. The reader would also be acquainted with the product’s External Interface Requirements which includes User Interfaces and Hardware Interfaces that the product has to offer. We will also delve into the the detail of the User Interface Design and will provide a refinement of the non-functional requirements that were proposed in the Software Requirements Specification document.

## 2.2 Product Scope

In this project our goal is to create a unified platform for campus related notifications for academic communities, as opposed to the numerous other platforms currently under use. Outlook’s Students’ Events’ uses a generic email to send all sorts of unsolicited information and is not a favorable platform for keeping track of events. Emails are not classified into categories. Facebook, another common medium for student activities also lacks organization and is often hard to navigate. For the case of the LUMS community, ‘The LUMS Discussion Forum’ on Facebook is always flooded with posts about seminars, trips, course and book swaps and it often takes some time to find the desired item despite the search option.

We introduce the concept of a “Notice Board” that will allow students to communicate among each other more effectively. The product will organise useful and relevant information scattered across all the above mentioned platforms into a single, unified platform, thus providing ready access to more organized information about everything campus related. The idea is inspired by a real noticeboard for schools that holds important information about the activities taking place in the institution. Each student will have access to the noticeboard and can use it to communicate to the rest of the LUMS community. Furthermore, the product will be free and restricted to the use for the LUMS community therefore providing a more dedicated environment for users. It will also minimize time wastage and improve communication efficiency.

## 2.3 Intended Audience and Document Overview

The Software Design Specification Document is intended for our clients and customers who will be members of the LUMS community belonging to diverse groups as:

- Students/Alumni
- Admininistration
- Faculty

This document is categorized into six different sections, each serving as an understanding of the project from a different perspective. The first section, Change Log, scans the idea of our project briefly, discussing the changes made in the development, from what was discussed in the SRS document. This is followed by the introduction of the project, which covers the purpose of the document, the scope of the project, summary of the document, definitions and abbreviations used in the document and the references and acknowledgments.The next section goes further into the functioning of the system, describing the constraints of the system, and the strategies used in the architecture of the system. The system architecture is then elaborated in the fourth section, which examines the organizational structure of the system, its decomposition into parts, their connectivity, interaction between subsystems, and the design of the NoticeBoard system. The User Interface Design follows next, describing the front end of the system. With images attached in this section, the readers of the document are able to get a better idea of what the system looks like and it’s operability. Following this, the non functional requirements discuss how the systems performance would be improved, the security measures that would be implemented for the system to be made secure, and the software attributes that make the system distinct.

## 2.4 Definitions, Acronyms and Abbreviations

| **Word** | **Meaning** |
|---|---|
| Admin/ Administration | The LUMS Administration |
| API | Application programming Interface |
| CSO | Career services office |
| Database | A database is a data structure that stores organized information. Most databases contain multiple tables, which may each include several different fields. For example, a company database may include tables for products, employees, and financial records. Each of these tables would have different fields that are relevant to the information stored in the table. |

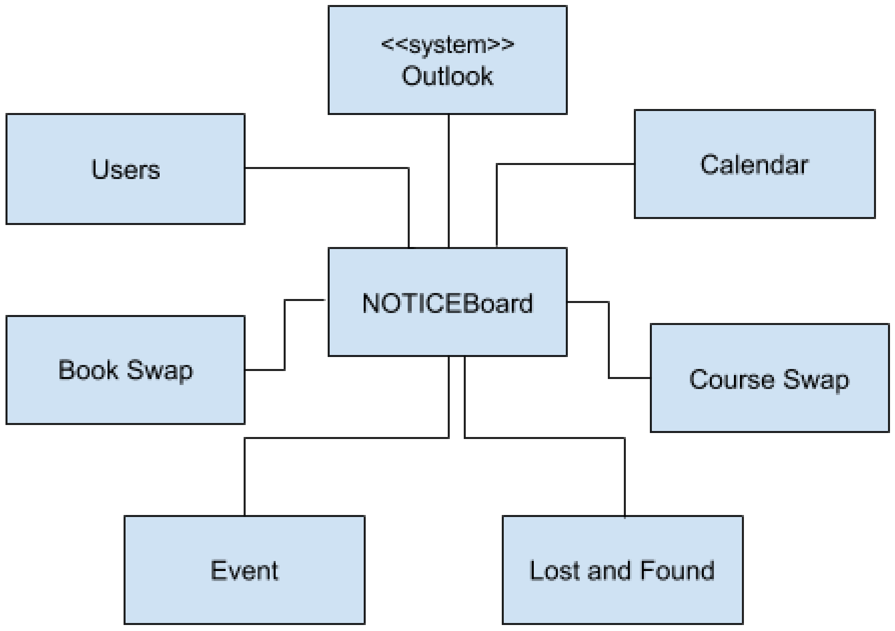
| | |
|---|---|
| Interface | The term "interface" can refer to either a hardware connection or a user interface. It can also be used as a verb, describing how two devices connect to each other. |
| Responsive Web Design | Responsive Web Design makes your web page look good on all devices (desktops, tablets, and phones). Responsive Web Design is about using CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen. |
| RESTful | Representational state transfer (REST). RESTful Web Services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. |
| Stakeholder | Any person or group of persons who are (or will be) affected by the system. |
| System | The product we are implementing |
| User(s) | Person who will use the website. |

## 2.5 References and Acknowledgments

*https://docs.angularjs.org/guide/introduction--AngularJS*
*https://en.wikipedia.org/wiki/Representational_state_transfer*
*https://en.wikipedia.org/wiki/Relational_model*
https://www.mongodb.com/mongodb-architecture

# 3 Overall Description

## 3.1 System overview



The context diagram drawn above represents the main system namely NOTICEBoard at the center, with no details of its interior structure, surrounded by all its interacting systems, environments and activities.

NOTICEBoard is a surface for posting notices like Event, Course/Book Swap and Lost & Found. Each User who creates this notice can edit and delete it, while others can view it.

Outlook by itself is a complete system and lies outside the system boundary, while the rest of the subsystems are actually part of the overall context of NOTICEBoard and are retrieved from the database. Outlook functions as a means of validating users on initial signup, since a mail is sent to the registering ID and in order for signup to complete, the user needs to login to Outlook and redirect to NOTICEBoard from an auto-generated mail sent on their Outlook accounts. Outlook IDs also provide uniqueness of valid users, since at-most one account can be made from the corresponding Outlook ID. Both these are part of our functional requirements as well.

The Users themselves are part of a separate subsystem since they are the main actors of the system context and are responsible for populating the rest of the subsystems with their operations such as Post, Delete, Edit etc. Once user-data is inserted into the respective subsystems, NOTICEBoard, the main entity in our context, retrieves and manipulates this data to provide the required functionality of this platform, with all the functional requirements. Functional Requirements which NOTICEBoard handles include managing user requests to viewing, updating and deleting data that they initially introduced into the system context.

Furthermore we can notify the users, through their own personal Calendar, in case if an upcoming event is being followed by the User. Hence all reminder notifications and user's Calendar instance management for the user's convenience are also part of this.

## 3.2 System constraints

On further analysis of the SRS for design, we realized some requirements entailed tedious constraints which needed some refinements in order to provide the required functionality with minimal compromise on the expected functionality. The reasons we were faced with system constraints are as follows:

- <u>Availability and Volatility of resources:</u> In order to sync events on NOTICEBoard with Google calendar's events, the user needs to provide Google ID and Password in order to allow Google's Calendar API to provide access to the relevant calendar. However, this entails an additional overhead of logging in to gmail in order to allow NOTICEBoard to access calendar. Firstly, the user might be reluctant to provide Google credentials on a separate entity not endorsed by Google. In addition to this, performance wise the user would essentially need to login twice on the same platform, which again becomes tedious for the user. Google credentials are therefore, subject to availability and become volatile after one session ends.
- <u>Standard Compliance and Security Requirements:</u> Using Google's Calendar API requires a validation process in which an API or OAuth2.0 key needs to be acquired for use of the API. While after the initial process, it secures the user's event data from google and all interactions with the calendar, however, the process deviates from proving user convenience. Instead of providing the user enhanced convenience, the user would first need to provide our platform their key so that the requisite setup could be done to initiate their separate instance of calendar on their NOTICEBoard accounts. So compliance with Google security requirements is a major issue that defeats the purpose of convenience in our platform and is beyond the scope of our product's focus.
- <u>Interoperability Requirements</u>: Google's calendar API is based on the RESTful, which means that the data it provides is in textual form. Once again, our platform's services includes the provision of a calendar interface which the user can view in a glance as opposed to a list of text which needs to be sifted through. In order to integrate the API data, a separate plugin for rendering calendar's interface would be required. Once again, operation is hampered due to the operability requirement of Google Calendar API.
- We faced several constraints in the design phase. The design choices in our case were greatly influenced by operations we could perform at the back end. So many design implementations that would have made the user experience better, and increased usability of the product were removed because they were very demanding for the frontend and backend development. An example of this is the removal of expandable cards for the showcasing of the features in the landing page. Additional tasks would have to be performed for fetching data or hiding data at the front end. In the industry the designer is much less influenced by the backend developers and works under lesser number of constraints.
- <u>Interface Requirements:</u> In order to deliver a minimalistic interface with all the functionality, we needed to Photoshop all the User Interface screens from scratch and then translate them into the respective CSS for use in HTML so that we could deliver our platform with custom organization of Events, Book/Course Swap and Lost and Found. Using off-the shelf components provided partially suitable interfaces. So designing the required interface also constrained us to using Adobe Photoshop.

## 3.3 Architectural strategies

We employed the following strategies in deciding the architectural design of our system:

- We intend to use AngularJS as our frontend. AngularJS is a structural framework for dynamic web apps. It allows using HTML as the template language and allows extending HTML's syntax to express the application's components clearly and succinctly. Not only this, but Angular Material provides code for different components that can easily be integrated in the interface of our platform to add visual appeal and a convenient application to use, which is one of our major functional requirements.
- We then incorporate our front end through NodeJS to our backend. We intend to use MongoDB for the backend. Having a layer of software between the data flow from the front end to the backend and vice versa, which in our case is Node JS, protects the database from direct communication with external interface. As discussed in the guest lecture by Munib, it is considered as a good practise to have a layer of software between the frontend and backend to prevent any corruption of data in the database, since this layer of software can sanitize requests that are made to the DB. Therefore, in our strategy we employ a three tier architecture, for efficient retrieval and writing of data to and from the database respectively.
- Using MongoDB as our backend provides us with a flexible data model. MongoDB's document data model makes it easy for you to store and combine data of any structure, without giving up sophisticated validation rules, data access and rich indexing functionality. We can dynamically modify the schema without downtime.
- We also chose a customizable calendar in order to render the user data for flagged (Interested/Going) Events onto the calendar as opposed to syncing with Google's Calendar API. The inconveniences of using Google's API, as mentioned in the previous section and the change log, are therefore circumvented by integrating a calendar plugin which not only fulfils the functional requirement of organizing campus related events in a visually pleasing and convenient manner but also provides ease of rendering separate instances of the calendar for each user.

All these frameworks work in perfect unison which allows us to design and create a very efficient web application.

# 4 System Architecture

## 4.1 System Architecture

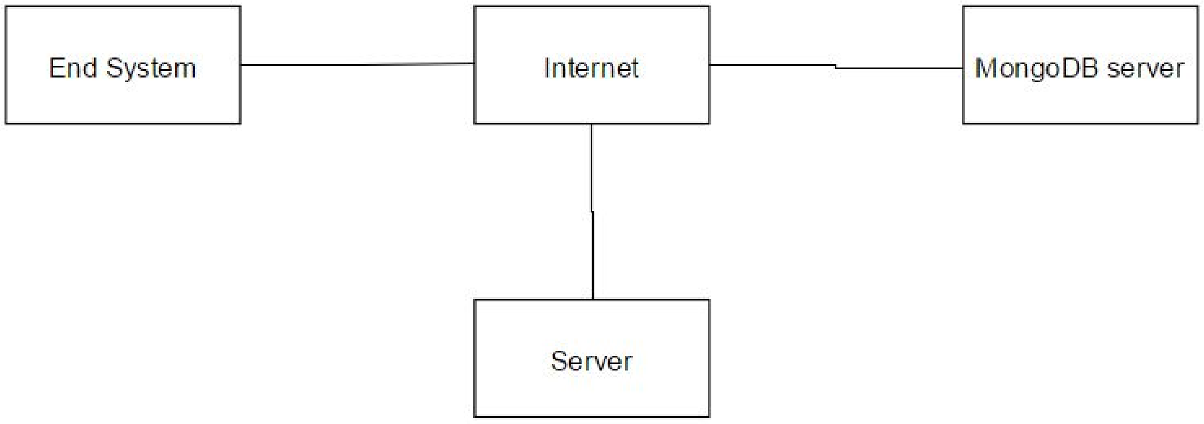
In deciding our architecture, we agreed to use the Client-Server pattern of existing web architectures. This proved to be the most suitable for our web application since we are providing a web service to the user over the internet. We categorize our system into subsystems which include the End System/Host (User), MongoDB server, and one main Server which is being controlled by a system administrator. All these subsystems are communicating over the internet as expected in a client-server architecture.

The End System/Host allows the user to make requests to our application. It contains a web browser that would be used to load the content from our servers and would present the response to user requests. The End System/Host can be a desktop, mobile device or any tablet since our application would be responsive.

The MongoDB server would be the hosting the database of our system. It would contain all user data that is posted by the user on our platform and that may be required to be presented to the user. Essentially, the MongoDB server would be the hub of our business logic since it contains the complete database and rules that would be defined in order to serve content to users.

Lastly, the main server (application server) would be responsible for handling all user requests and forwarding them to the MongoDB server in order to forward response to the user. It would allow clients to establish a connection to NoticeBoard if they are already registered over the application or would handle Sign Up requests for new users. It would also allow a connection to be established to MongoDB server in order to access the database. Having this layer between database and the End System/Host would prevent exposure of the DB to the wild and can prevent invalid accesses by sanitizing user requests. This server would be controlled by the system administrator.

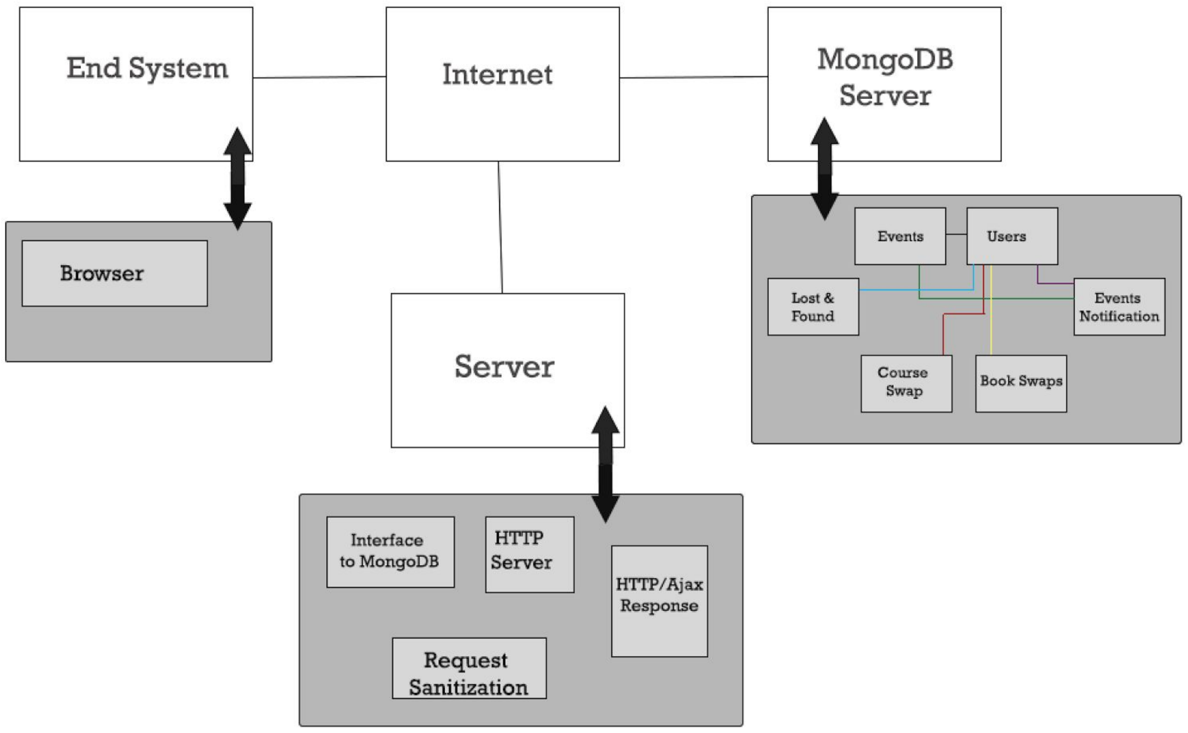
The following diagram provides an overview of this architecture.



The components within MongoDB are described as follows:

- The component 'Events' consists of a list of several upcoming events. Each event has a Name, Location, Description, Picture and a Timings component. These details allow the user to be able to decide whether to get further updates about the event or not. Once the user chooses to get further information and updates, he/she clicks on the 'Interested' button, assigned for each event, which then sets the events date and timings to the Calendar component of the individual user, and keeps notifying the user about any updates, via the notification component.
- The Lost & Found component provides users with a stage to contact other users who may have lost or found personal items on Campus. Since no such platform exists at the moment, people tend to flood Outlook via emails and LUMS Discussion Forum via posts. The system further provides details about the item lost or found along with the details of the person who has posted, making it easier to contact them.
- Book or course swapping subsystem allows details of the course or book wanted, by the individual posting and those, that need to be exchanged, to be displayed, to allow other users looking to swap to have easy access.
- All of the above use User component of MongoDB to relate different users with Event Notifications, Book/Course swap, Lost and Found and Events.

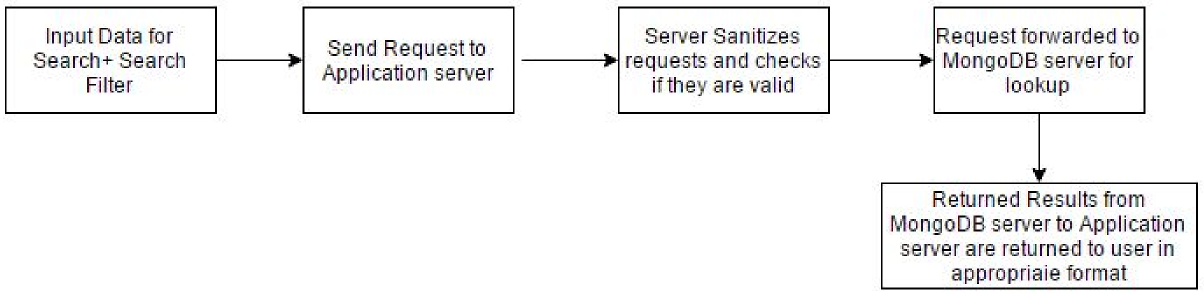
The component diagram is as follows:



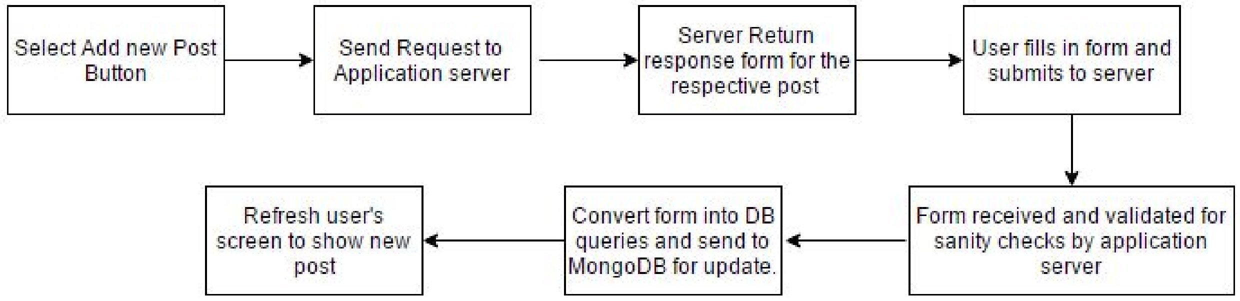
## 4.2 Subsystem Architecture

The sequence diagrams for use cases are as follows:

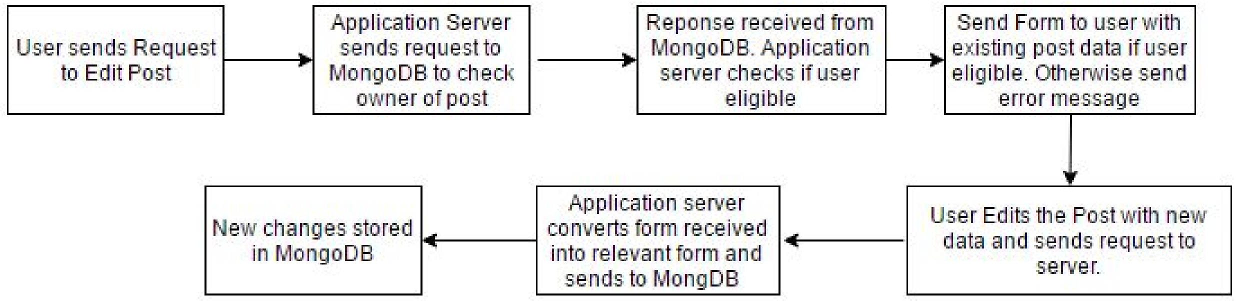
- <u>Search:</u>


Input Data for Search+ Search Filter
Send Request to Application server
Server Sanitizes requests and checks if they are valid
Request forwarded to MongoDB server for lookup
Returned Results from MongoDB server to Application server are returned to user in appropriaie format

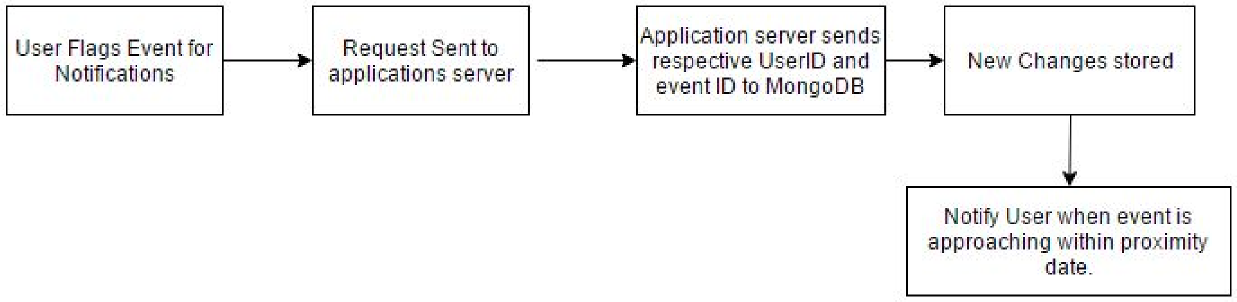
- <u>Post for course/book swap or events:</u>


Select Add new Post Button
Send Request to Application server
Server Return response form for the respective post
User fills in form and submits to server
Form received and validated for sanity checks by application server
Convert form into DB queries and send to MongoDB for update.
Refresh user's screen to show new post

- Editing Posts:


User sends Request to Edit Post
Application Server sends request to MongoDB to check owner of post
Reponse received from MongoDB. Application server checks if user eligible
Send Form to user with existing post data if user eligible. Otherwise send error message
User Edits the Post with new data and sends request to server.
Application server converts form received into relevant form and sends to MongDB
New changes stored in MongoDB

- Event Notifications:

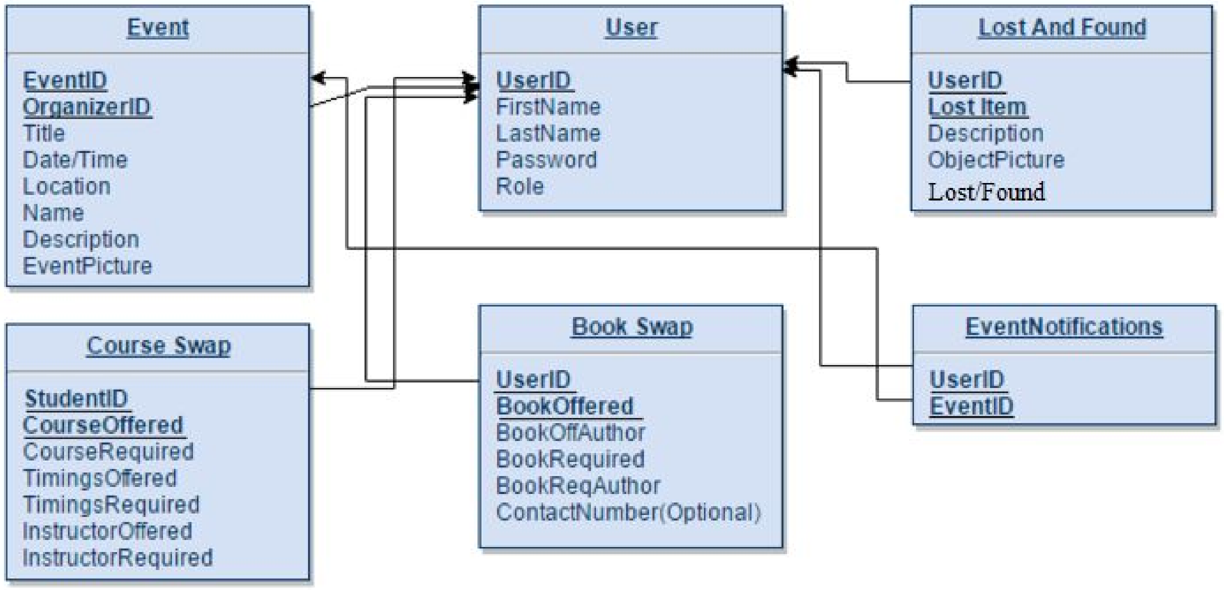

User Flags Event for Notifications
Request Sent to applications server
Application server sends respective UserID and event ID to MongoDB
New Changes stored
Notify User when event is approaching within proximity date.

## 4.3 Database Model

The Database Model we have chosen for our software is Embedded Data Model. MongoDB stores data as documents in a binary representation called BSON (Binary JSON). Embedded data models allow applications to store related pieces of information in the same database record. As a result, applications may need to issue fewer queries and updates to complete common operations. Documents that share a similar structure are typically organized as collections. Collections are analogous to a table in a relational database: documents are similar to rows, and fields are similar to columns. MongoDB documents tend to have all data for a given record in a single document, whereas in a relational database information for a given record is usually spread across many tables. The database would consist of 6 tables with their respective Primary key and Referential Integrity constraints which are described in the following subsections.

### 4.3.1 Database schema and detailed description

**Relational model database schema:** (the underlined fields are primary key)
Note: The arrowhead shows that the pointed table is referenced by the table that is pointing to it. (to clarify referential integrity constraints and primary-foreign key logic).



| Tables | Relationship |
|---|---|
| User -> Event | One to many |
| User -> Course Swap | One to many |
| User -> Book Swap | One to many |
| User -> Lost and Found | One to many |
| Event -> Event Notifications | One to many |
| User -> Event Notifications | One to Many |

| Tables | Fields | Description |
|---|---|---|
| User | • <u>UserID</u><br>• First Name<br>• Last Name<br>• Password<br>• City<br>• Role | • In order to register valid users to the website and to keep track of their activities like view/edit/create/delete post, we need to store user's information in a table.<br>• **Constraint:** Here <u>UserID</u> acts as a **primary key** prevents the same user from making multiple accounts on the same ID. |
| Event | • <u>Event ID</u><br>• <u>Organizer ID</u><br>• Name<br>• Title<br>• Location<br>• Date & Time<br>• Picture<br>• Description | • Given that user has registered, user can now permissibly create/edit/delete/view event notice. These functionalities can only be implemented if we have saved event instances in a collection with unique Auto-Increment Event IDs.<br>• **Constraint:** In case of Event, both <u>Event ID</u> and <u>Organizer ID</u> are taken to be primary key that determines a particular event. This combination allows a user to post multiple events without any duplication of the same event by the same user.<br>• **Referential integrity:** <u>Organizer ID</u> acts as a foreign key value that references a valid, existing primary key in the parent table, which in this case is UserID in the User table. This would allow NOTICEboard to relate each event with a unique user without having to duplicate the information. |
| Course Swap | • <u>StudentID</u><br>• <u>Course Required</u><br>• <u>Course Offered</u><br>• Instructor Offered<br>• Instructor Required<br>• Timings Offered<br>• Timings Required | • Once logged in, user can choose to view/create/edit/delete course swap post. Assuming Microsoft Outlook to be the default mail client, User can press a mailto link to open a new mail window with auto generated mail fields that can be sent to the desired person. But this can only be possible if we keep record of Course Swap posts in a table.<br>• **Constraint:** Course Swap also has multiple attributes comprising **primary key**, that uniquely identifies tuples, including <u>StudentID</u>, <u>Course Offered</u> and <u>Course Required.</u> This combination would allow students to request for swaps without having to repost for the same course since the primary key constraint would forbid the transaction.<br>• **Referential integrity:** <u>Student ID</u> acts as a foreign key value that references a valid, existing <u>UserID</u> in User table. |
| Book Swap | • <u>UserID</u><br>• <u>Book Offered</u><br>• Book Off Author<br>• Book Required | • After signing in user can choose to view/create/edit/delete course book post. Assuming Microsoft Outlook to be the default mail client, User can press a mailto link to open a new mail window with auto generated mail fields that can be sent to the desired person. But this can only be possible if we keep record of Book posts in a table with this combination of primary keys as described for the rest of |

| | | |
|---|---|---|
| | • Book Req Author<br>• Contact Number(optional) | the tables.<br>• **Constraint:** Similarly a Book Swap post can be identified using <u>UserID</u> and <u>Book offered</u> as **primary key.**<br>• **Referential integrity:** <u>UserID</u> acts as a foreign key value that references a valid, existing <u>UserID</u> in the User table. |
| Lost & Found | • <u>UserID</u><br>• <u>Lost Item</u><br>• Description<br>• Contact Number(optional) | • Once logged in, user can choose to view/create/edit/delete Lost & found post. Assuming Microsoft Outlook to be the default mail client, User can press a mailto link to open a new mail window with auto generated mail fields that can be sent to the desired person. But this can only be possible if we keep record of Lost & Found posts in a table with this primary key.<br>• **Constraint:** <u>UserID</u> and <u>Lost Item</u> are combination of attributes chosen as a **primary key** to find the Lost & Found item in order to prevent duplication of post for the same item by the same user.<br>• **Referential integrity:** <u>UserID</u> acts as a foreign key value that references a valid, existing <u>UserID</u> in the User table. |
| Event Notification | • <u>User ID</u><br>• <u>Event ID</u> | • User can receive Event Notifications for events marked as going in case there is an update or edit to the event. Also send a reminder a period of time (user defined) before the event. This can be achieved by storing attributes like User ID and Event ID in Event Notification table notifications to the user and rendering his calendar instance.<br>• **Constraint:** While Event notification is comprised of only 2 attributes namely <u>Event ID</u> and <u>User ID</u>, and both of them are **primary keys**. The would allow us to keep track of what events each user is interested in and keep an organized record of what people are going to a particular event. This would help for providing information to the calendar plugin.<br>• **Referential integrity:** <u>UserID</u> and <u>EventID</u> act as a foreign key value. Here <u>UserID</u> references <u>UserID</u> in the User table while <u>EventID</u> points <u>EventID</u> in the Events table. Both these foreign keys can be used to reference event and user information for manipulation with minimal duplication. |

### 4.3.2 Database

- MongoDB enables developers to design and evolve the schema through an iterative and agile approach, while enforcing data governance.
- **Integrated Feature Set.** Analytics and data visualization, text search, graph processing, geospatial, in-memory performance and global replication allow you to deliver a wide variety of real-time applications on one technology, reliably and securely. RDBMS systems require additional, complex technologies demanding separate integration overhead and expense to do this well.
- **Flexible Data Model.** MongoDB's document data model makes it easy for you to store and combine data of any structure, without giving up sophisticated validation rules, data access and rich indexing functionality. You can dynamically modify the schema without downtime. You spend less time prepping your data for the database, and more time putting your data to work.
- **Expressive query language & secondary Indexes.** Users should be able to access and manipulate their data in sophisticated ways to support both operational and analytical applications. Indexes play a critical role in providing efficient access to data, supported natively by the database rather than maintained in application code.
- **Multi-Datacenter Scalability.** MongoDB can be scaled within and across geographically distributed data centers, providing new levels of availability and scalability. As your deployments grow in terms of data volume and throughput, MongoDB scales easily with no downtime, and without changing your application. And as your availability and recovery goals evolve, MongoDB lets you adapt flexibly, across data centers, with tunable consistency.
- Leverage data and technology to maximize competitive advantage
- Reduce risk for mission-critical deployments
- Accelerate time-to-value
- Dramatically lower total cost of ownership

## 4.4 External Interface Requirements

### 4.4.1 User Interfaces

- GUI will be used as the interface. There will be no command line interface.
- We are designing for the majority of the student population; the population which is colorblind are assumed to find the interface more difficult to use, but no major functionality of the product will depend on the perception of colors, allowing colorblind users to still interact with the interface effectively.
- Optimizing the interface for users can be done by organizing contents of the webpage in a way that allows most recent items first.

- User Interface Design should be kept as user-friendly as possible. Our goal is to provide simple, straightforward and easy to use UI.
- When user opens the website application, he/she can either register or sign in. First page is comprised of a sign up button and fields for user credentials
- Through sign up section on the landing page, users can get themselves registered by filling in the required fields such as email, full name and password.
- Home page is designed in such a way that it provides an overview of all the subsystems. The user can either browse through the events section, swapping section or lost and found or view his/her calendar.
- Each of the divisions on the main page is a hub where users can post, view, search and delete post(s) for the respective categories.

### 4.4.2 Hardware Interfaces

The webpage will not use any dedicated hardware on the user side. The database will connect to the web interface through the operating system of the user. There will be server hardware that will be used to manage information.

# 5 User Interface Design

## 5.1 Description of the user interface

Front-end tool:
- Adobe Dreamweaver (IDE)
- Adobe Photoshop

Front-end technologies:
- HTML 5, CSS 3
- JavaScript/JQuery
- AngularJS

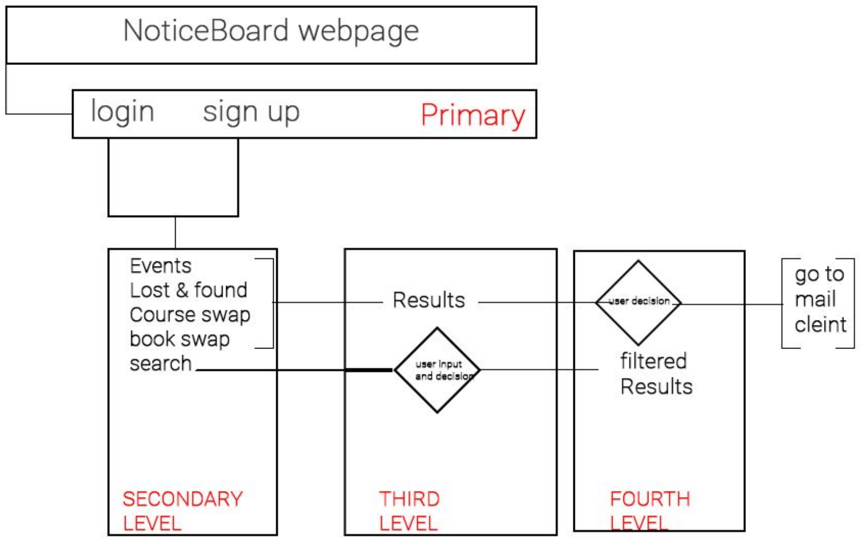
Adobe Dreamweaver provides ease in transitioning from a UI design in Adobe Photoshop to design implementation in HTML and CSS. The Extract tool will hint about CSS properties for the layers created in Photoshop, which can be imported in this extract tool. The extract tool also provides extraction of graphical assets (such as photos, logos, thumbnails) from the layers and importing on to img tags in the html.

Adobe Photoshop allows flexibility and high customizability when it comes to designing the screens. The layering functionality allows to create and group UI elements according to our intended design.

A suite of front-end technologies were used in this project from the familiar HTML, CSS to JavaScript and JQuery for dynamic elements and interaction with page elements. AngularJS is used to generate the feed for our subsystems (events, lost and found and swaps) and to power our search bar and the underlying search functionality.

The presentation or view layer of the MVC architecture in this case would handle dynamic page generation for the main page plus dynamic feed generation for the various subsystems (events. swaps and lost and found), forms management for the new post subscreens, search result display for queried posts, notifications for reminders and updates and dynamic rendering of calendar to reflect any changes in a user' calendar (such as when a user favorites/subscribes to an event).

## 5.2 Information architecture



There are a total of two screens in the service, with a number of subscreens or pop-ups in between.

The first screen is the landing page which serve as an entry point into the service, either through login or sign-up. This screen also introduces potential user to the service, highlighting a few major features and also introduces the team behind the service.

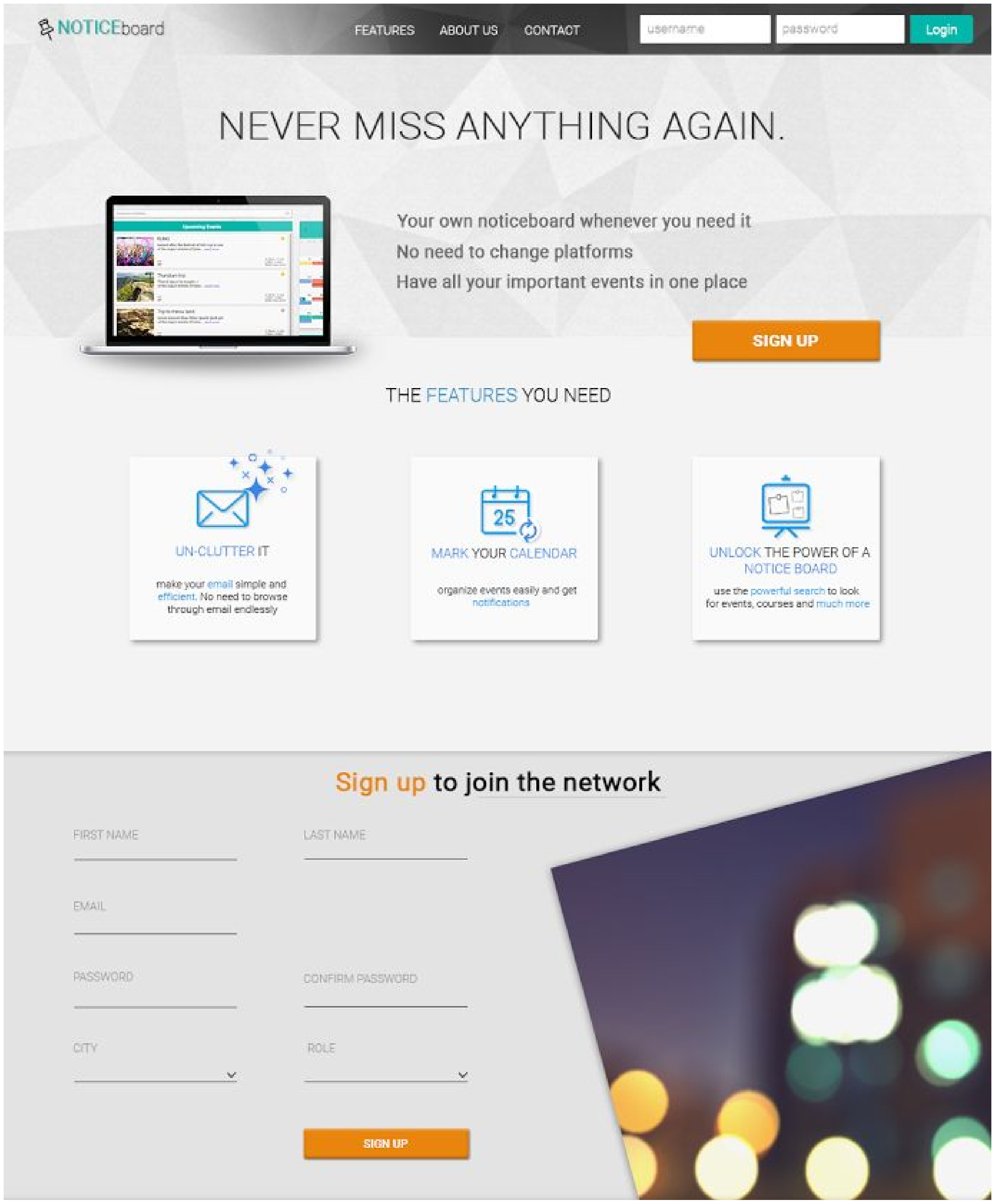
The second screen is where the magic happens. It is the place where the core of our service resides, with the entire MVC found here. A number of subsystems (such as calendar, events, lost and found and course/book swaps) are located here. This is where new posts are created through user interaction and input to controller, which modify the model and where posts are viewed through the presentation layer.

Numerous components are also housed here, with the search bar for search functionality across various post types, a notification shade for viewing recent notifications and a profile dropdown for seeing who’s logged into the service currently.

A number of pop-ups or as we would say sub-screens can also be found on this screen. Creating a new post for any type will open a pop-up (or a sub-screen) specific to the post type with a form to fill and submit to create a new post entry in the database on the backend. Also, the notification shade is itself a subscreen to view notifications such as event reminders or change in venues etc.

Clicking on a post will open another subscreen (i.e. a popup) that will display the detailed post together with option to edit or delete the post if it’s of the user viewing.

## 5.3 Screens



NEVER MISS ANYTHING AGAIN.

Your own noticeboard whenever you need it
No need to change platforms
Have all your important events in one place

THE FEATURES YOU NEED

UN-CLUTTER IT
make your email simple and efficient. No need to browse through email endlessly

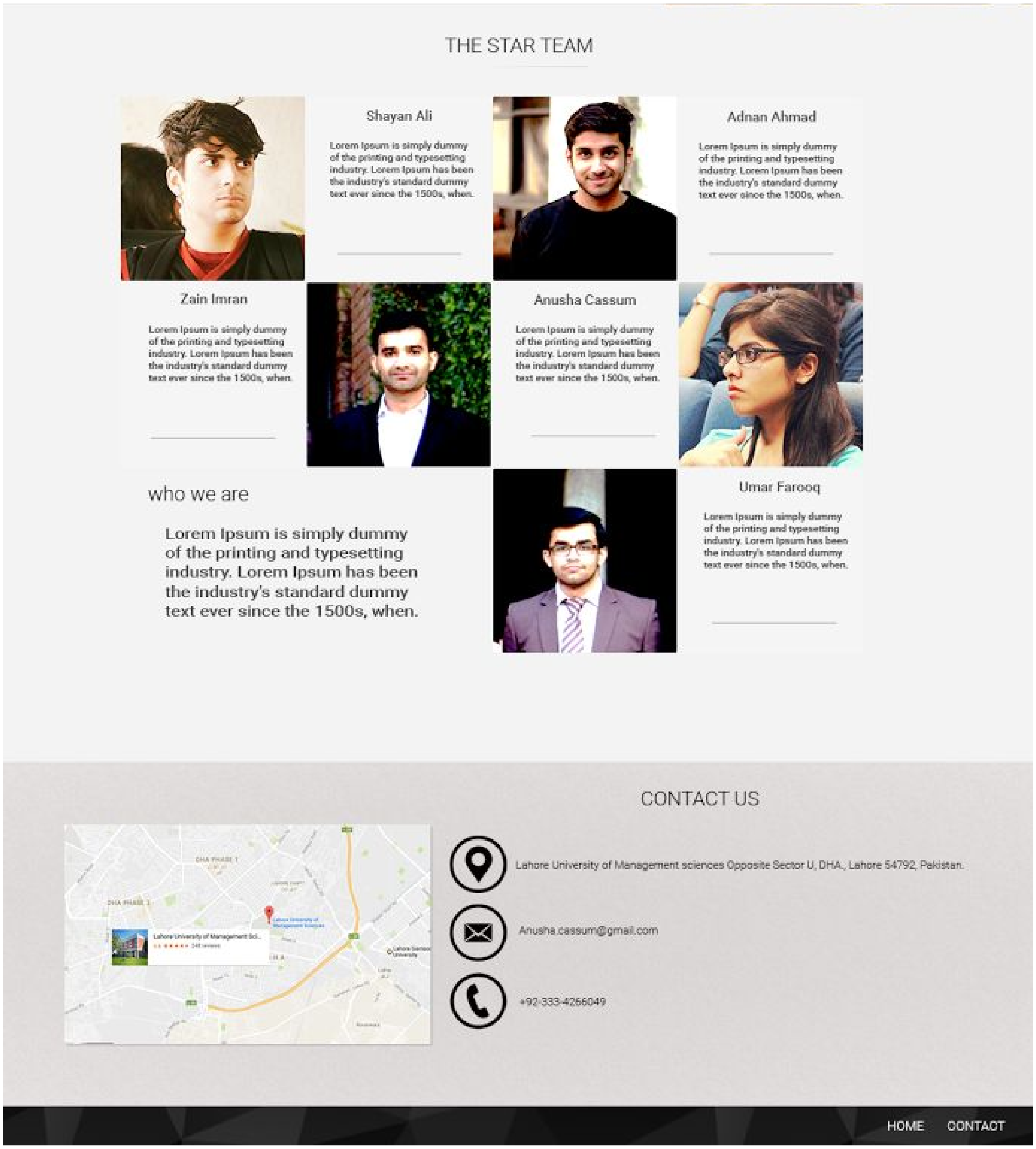
MARK YOUR CALENDAR
organize events easily and get notifications

UNLOCK THE POWER OF A NOTICE BOARD
use the powerful search to look for events, courses and much more

Sign up to join the network

FIRST NAME
LAST NAME
EMAIL
PASSWORD
CONFIRM PASSWORD
CITY
ROLE

*Landing page*



THE STAR TEAM

Shayan Ali

Lorem Ipsum is simply dummy of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when.

Adnan Ahmad

Lorem Ipsum is simply dummy of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when.

Zain Imran

Lorem Ipsum is simply dummy of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when.

Anusha Cassum

Lorem Ipsum is simply dummy of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when.

who we are

Lorem Ipsum is simply dummy of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when.

Umar Farooq

Lorem Ipsum is simply dummy of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when.

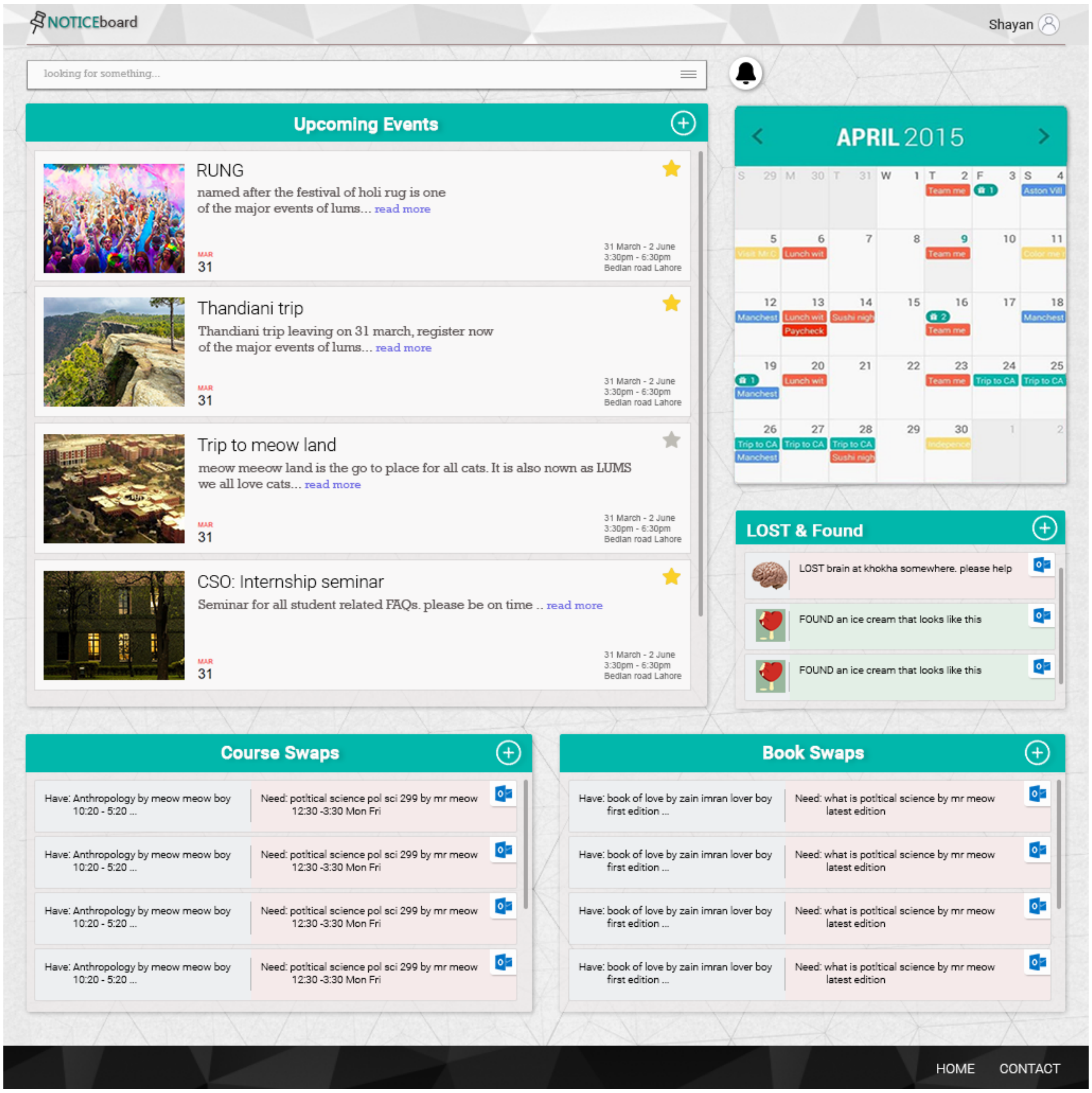
CONTACT US

Lahore University of Management sciences Opposite Sector U, DHA., Lahore 54792, Pakistan.

Anusha.cassum@gmail.com

+92-333-4266049

*Landing page (cont…)*



**Upcoming Events**

RUNG
named after the festival of holi rug is one of the major events of lums... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

Thandiani trip
Thandiani trip leaving on 31 march, register now of the major events of lums... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

Trip to meow land
meow meeow land is the go to place for all cats. It is also nown as LUMS we all love cats... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

CSO: Internship seminar
Seminar for all student related FAQs. please be on time .. read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

**APRIL 2015**

| S | M | T | W | T | F | S |
|---|---|---|---|---|---|---|
| 29 | 30 | 31 | 1 | 2 Team me | 3 1 | 4 Aston Vill |
| 5 Visit Mr.C | 6 Lunch wit | 7 | 8 | 9 Team me | 10 | 11 Color me |
| 12 Manchest | 13 Lunch wit Paycheck | 14 Sushi nigh | 15 | 16 2 Team me | 17 | 18 Manchest |
| 19 1 Manchest | 20 Lunch wit | 21 | 22 | 23 Team me | 24 Trip to CA | 25 Trip to CA |
| 26 Trip to CA Manchest | 27 Trip to CA | 28 Trip to CA Sushi nigh | 29 | 30 Indepence | 1 | 2 |

**LOST & Found**

LOST brain at khokha somewhere. please help

FOUND an ice cream that looks like this

FOUND an ice cream that looks like this

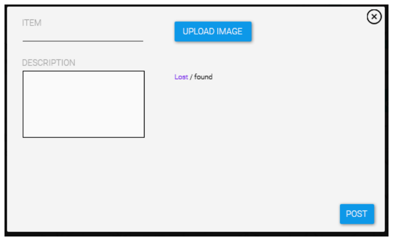
**Course Swaps**

| | |
|---|---|
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: poltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: poltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: poltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: poltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |

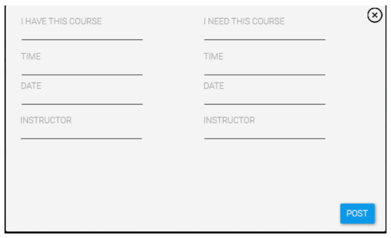
**Book Swaps**

| | |
|---|---|
| Have: book of love by zain imran lover boy first edition ... | Need: what is poltical science by mr meow latest edition |
| Have: book of love by zain imran lover boy first edition ... | Need: what is poltical science by mr meow latest edition |
| Have: book of love by zain imran lover boy first edition ... | Need: what is poltical science by mr meow latest edition |
| Have: book of love by zain imran lover boy first edition ... | Need: what is poltical science by mr meow latest edition |

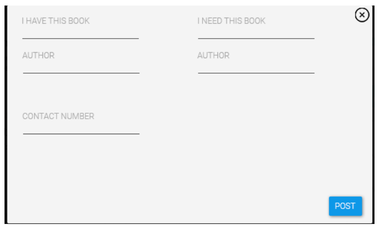
*Main page*



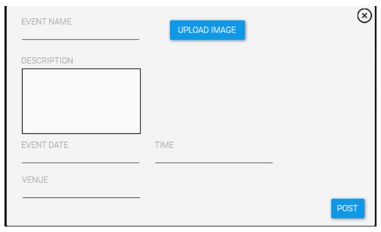
*Lost and found post*



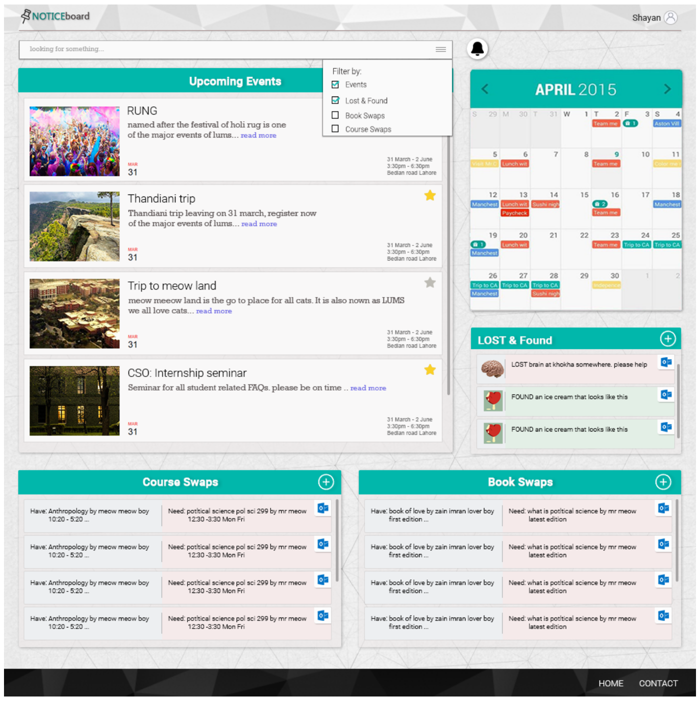
*Course swap post*



*Book swap post*



*Event post*



RUNG
named after the festival of holi rug is one
of the major events of lums... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

Thandiani trip
Thandiani trip leaving on 31 march, register now
of the major events of lums... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

Trip to meow land
meow meeow land is the go to place for all cats. It is also nown as LUMS
we all love cats... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

CSO: Internship seminar
Seminar for all student related FAQs. please be on time .. read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

APRIL 2015

| S | M | T | W | T | F | S |
|---|---|---|---|---|---|---|
| 29 | 30 | 31 | 1 | 2 Team me | 3 🗓 1 | 4 Aston Vill |
| 5 Visit Mr.C | 6 Lunch wit | 7 | 8 | 9 Team me | 10 | 11 Color me |
| 12 Manchest | 13 Lunch wit Paycheck | 14 Sushi nigh | 15 | 16 🗓 2 Team me | 17 | 18 Manchest |
| 19 🗓 1 Manchest | 20 Lunch wit | 21 | 22 | 23 Team me | 24 Trip to CA | 25 Trip to CA |
| 26 Trip to CA Manchest | 27 Trip to CA | 28 Trip to CA Sushi nigh | 29 | 30 Indepence | 1 | 2 |

LOST & Found
LOST brain at khokha somewhere. please help
FOUND an ice cream that looks like this
FOUND an ice cream that looks like this

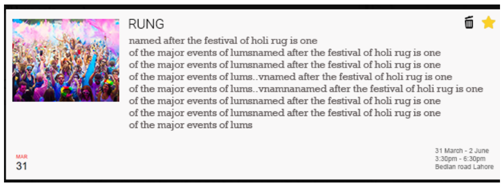
Course Swaps

| | |
|---|---|
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: potltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: potltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: potltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |
| Have: Anthropology by meow meow boy 10:20 - 5:20 ... | Need: potltical science pol sci 299 by mr meow 12:30 -3:30 Mon Fri |

Book Swaps

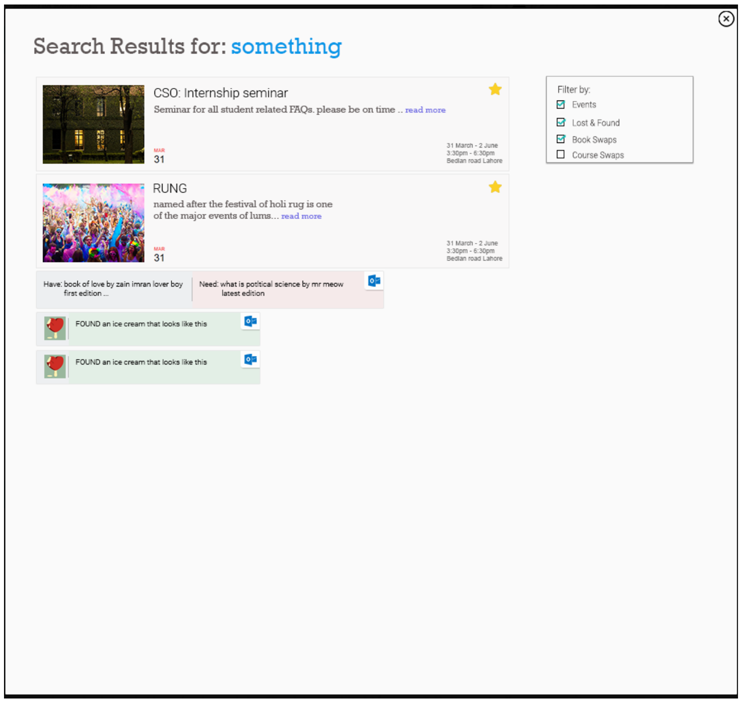
| | |
|---|---|
| Have: book of love by zain imran lover boy first edition ... | Need: what is potltical science by mr meow latest edition |
| Have: book of love by zain imran lover boy first edition ... | Need: what is potltical science by mr meow latest edition |
| Have: book of love by zain imran lover boy first edition ... | Need: what is potltical science by mr meow latest edition |
| Have: book of love by zain imran lover boy first edition ... | Need: what is potltical science by mr meow latest edition |

*Filtering by categories in search- accessible via the hamburger button*



RUNG
named after the festival of holi rug is one
of the major events of lumsnamed after the festival of holi rug is one
of the major events of lumsnamed after the festival of holi rug is one
of the major events of lums..vnamed after the festival of holi rug is one
of the major events of lums..vnamnanamed after the festival of holi rug is one
of the major events of lumsnamed after the festival of holi rug is one
of the major events of lumsnamed after the festival of holi rug is one
of the major events of lums
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

*Expanded event post*


Search Results for: something

CSO: Internship seminar
Seminar for all student related FAQs. please be on time .. read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

RUNG
named after the festival of holi rug is one of the major events of lums... read more
MAR 31
31 March - 2 June
3:30pm - 6:30pm
Bedian road Lahore

Have: book of love by zain imran lover boy first edition ...
Need: what is potitical science by mr meow latest edition

FOUND an ice cream that looks like this

FOUND an ice cream that looks like this

Filter by:
- [✓] Events
- [✓] Lost & Found
- [✓] Book Swaps
- [ ] Course Swaps

*Search results*

LANDING PAGE:

The landing page has a header at the top which has the username and password fields for logging in on the top right. Next to that are anchor links to various sections of the landing page, such as features, about us and contact. Clicking any of these links will scroll the landing page to that section of the page and bring it to viewport.

In the top viewport, there is a quick mention of the three features of the service, together with a sign up button above it. The sign up button is itself an anchor link which will take the user to the second viewport of the landing page i.e. the signup form.
On a viewport below is the mention of the team behind the service and scrolling a bit down will reveal the contact info along with a map location of the team in an embedded google maps. On the very bottom is a footer with anchor links to Home and Contact.

MAIN PAGE:

The main page is divided into four sections, one for each main feature. A drop down menu on the top right hand side of the page shows the user's email ID and profile picture, and gives the option to logout of NoticeBoard, by clicking on the "Sign out" button.

The search bar is an input field positioned on the top of the page, right below the NoticeBoard header. It requires the user to type whatever the user is searching for on the homepage. Through a dropdown-menu right next to the search bar, the user will be given the choice to search in either the events, lost and found or course/book swap and on hitting enter the search query will be submitted to the backend.

Right next to the search bar is a bell icon which upon clicking will open a pop-up displaying all recent notifications. These notifications can be about a subscribed event for example. The bell icon will change color on receiving a new notification.

On the left of the homepage is the Upcoming Events section, with infinite scrolling to show all events arranged according to most recent. By further clicking on any event post, the detailed post for that event will pop up, showing the event's full description. With the rest of the screen darkened, the user is able to focus on that specific post. On the header of this section is plus button which will open a pop-up to add new post.

On the right of the Upcoming Events page is the calendar, which will display all events subscribed by the user. Just below the Calendar, is the Lost and Found Section, with infinite scrolling to show all lost and found posts arranged according to most recent. By further clicking on any lost and found post, the detailed post will pop up, showing the post's full description along with the details of the individual who had posted, for the user to be able to contact the individual. On the header of this section is plus button which will open a pop-up to add new post.

Just below the Upcoming Events section is the Course Swapping Section  showing all the posts in an infinite scroll fashion. To the right of this section is the Book Swap Section, also showing all its posts in an infinite scroll fashion. Clicking on a post in any of these sections opens the full post, where further details of the post are displayed. On the header of this section is plus button which will open a pop-up to add new post.

On the very bottom is a footer with links to Home, Contact and Terms of Service pages.

## 5.4 User interface design rules

One of our main requirements was to enhance the user experience and have good usability. The basic structure of the webpage has been kept simple. The following 'rules' have been considered during the design of the interface. These are also known as principles of graphic design. The aim was to make the interface approachable so users do not find it difficult to switch across platforms and making the design immediate so that a user does not have to go through multiple steps in order to accomplish a task. This goes hand in hand with the rest of the development, as multi step processes at the design phase will also incur addition effort in frontend and backend development.

- Simplicity: The design was made as simple as possible. Reduction of extra information and operations made the design simpler, this conflicted against the visibility of the interface in several situations.

- Visibility: this relates to the operations that can be performed by the user being available to the user. We made the functions of the system visible by making the interface minimalistic where possible. Some functions lack visibility such as: the cards for posts lack indication that the perform an action when clicked on.
- Feedback: Our design was reduced in the context of feedback, because of constraints of the backend side. But many features do have proper feedback such as buttons from angular material that already supports feedback.
- Consistency:
  - Fonts: the font ‘Roboto’ has been used consistently with different styles eg. ‘light’, ‘thin’, ‘regular’ and ‘bold’
  - Colors: The color scheme used here has shades of grey and blue.
  - Theme: A paper-like pattern has been used for headers to connect with the concept of a notice board
  - Existing interfaces: We have followed many conventions of web design such as login and signup sections. This makes it easier for users to communicate with the interface.
- Balance: Elements have been balanced w.r.t the vertical axis.
- Reading Flow: Web pages have been designed to support reading flow. At some places this was not possible and there were conflicts with consistency. for example the interested button could have be placed in right bottom side of the event card (in the main page) to allow users to mark as interested after looking through the event ,but this is not consistent with the external conventions of design for websites. So the choice to put the button on the top right corner was made instead as this seemed a more logical.
- White space: has been utilized to make separation between different elements of the webpage. Lines have only been used to make the design aesthetically pleasing.
- Grouping: This is seen where different elements of a post haven been grouped inside a structure to make into one object. For example all the elements of an event have been bounded by a rectangle to make it into one event. And all the events have been grouped into another larger rectangle to make it into the events section.
- Contrast: This principle has been employed at the landing page to attract attention to the sign up button. This is done by the use of the orange color which stands out against the blue and white interface. The color of the text in the sign up section also creates a contrast as described.

# 6 Other Non-functional Requirements

## 6.1 Performance Requirements

- Since we are developing a Web Application, it is necessary that latency is minimized and users are delivered dynamic content with minimum delay. The new page is expected to load within 4 seconds.
- Security is also another non-functional requirement. Since our primary focus is to develop a *NoticeBoard* platform for users, user must be able to rely on our platform. For users to be able to trust our platform, we must provide them with an assurity of security. A virus-prone or malware infected site would not receive any response from users and any risk of private information compromise would discourage users to use this facility at all. Due to different constraints, a fully secure website will not be implemented, but basic security features will be implemented. One of these security features include verification of email ID when signing up for the first time, ensuring that the ID belongs to a valid user. Furthermore, to prevent the possibility of hacking another person's account, there will be specific requirements for the password, such as Uppercase letters, numbers, thereby increasing the strength of the password, making it less likely to be stolen.
- Since the main motivation for users to use our platform is its effectiveness in terms of time consumption, we need to ensure our search engine and live feed is highly efficient and filters results much more quickly as compared to sifting through the spammed and cluttered emails. Like minimum latency, efficient search is also a non-functional constraint for our platform to serve its purpose and to encourage people to use our platform to save their time.
- Another performance requirement is the number of online users that can be served at any given time. If the system is incapable of supporting a reasonable number of people at peak hours, then many users would face lag which would directly nullify the purpose of the platform to save time and bring more organization.
- Fault Tolerance: This is self explanatory as there should be a backup in case there is server crash at peak hours or due to power outage, hence data must be backed up at regular intervals in order to provide consistency in such scenarios.

## 6.2 Safety and Security Requirements

Some of the security requirements that we found are expected by the client are as follows:

- In order to keep track of the fact that only eligible people are accessing the platform and posting information, a login/signup would first be mandated so that only people of LUMS are on the platform. Any outsider trying to make an account would simply be denied since only people with a valid LUMS email ID would be able to sign up. Verification would prevent any mischief.
- Another safety requirement is that a valid user email ID can only make one account. This would prevent bogus accounts from being formed and would ensure that there is no identity misuse.

- A user would not be able to post anonymously. This would be done so that there is complete transparency and everyone could be cognizant of what announcement has been made by anyone else without there being any room for fraud.
- When setting up password, the user would be required to use Uppercase letters and numbers to increase the strength of the password, reducing the likeliness of hacking.
- Passwords would be stored as hashes on the server side so even our server is compromised, the passwords would not leak.
- The login/signup page would be in https so as to provide a secure socket connection

## 6.3 Software Quality Attributes

**Reliability**

- System should give correct results for searches 90% of the time. Correct results are defined as results that are relevant to the input.
- System backend should not be offline for more than 1 day.
- This will be done by integrating efficient and advanced algorithm that filter search results more reliably.

**Maintainability**

- Code for the product should have enough readability and enough abstraction to allow further developments and changes that come in later stages of development. Since these attributes cannot be measured quantitatively they will be measured qualitatively.
- The system should be regularly recycled off garbage data such as session data, in order to make storage more efficient and enhance performance.

**Testability:**

- The program code should be tested for any memory leaks in order to prevent memory wastage and improve system performance.
- Core features and components should be tested individually before being integrated with the whole system.
- Before any new code is integrated into the system, it should thoroughly be tested by a known set of test cases in a separate environment before it is included into the main code. This ensures that the code remains bug free and any new bugs are introduced by new code most likely as opposed to existing code.

**Availability:**

- Availability means that the system is operational when it is used
- Measuring the average system availability as a scale for measuring availability
- Measurements obtained from 100 hour of full usage by a significant number of test users during testing.
- Target: More than 96% of time
- Wish: 100% of time

**Robustness:**

- Appropriate error handling must be done in order to cater for rogue usage of the platform.

- The platform should restore to normal state on refreshing the page in case of an erroneous gateway request.

**Security:**

- The login/signup pages should be coded in HTTPS/SSL so that the authentication process is secured.
- Passwords would be stored as hashes on the server side so even our server is compromised, the passwords would not leak.
- When setting up password, the user would be required to use Uppercase letters and numbers to increase the strength of the password, reducing the likeliness of hacking.

**Interoperability / Portability**:

- The system should work lag free on all web browsers, independent of the operating system.

## Appendix B – Contribution Statement

| *Name* | *Contributions in this phase* | *Approx. Number of hours* | *Remarks* |
|---|---|---|---|
| *Adnan Ahmed* | System Architecture | *10-12* | |
| *Anusha Cassum* | System Architecture<br>User Interface Design | *10-12* | |
| *Shayan Ali* | User Interface Design | *10-12* | |
| *Umar Farooq* | Overall Description<br>System Architecture<br>Other Non-functional Requirements | *10-12* | |
| *Zain Imran* | User Interface Design<br>Other Non-functional Requirements | *10-12* | |