Campus Talk - Project Part 4

Team: Aadish Gupta
Pallavi Madasu
Yogitha Mahadasu

Title: Campus Talk

Project Summary: Platform for all the campus students to collaborate and keep up to date with latest events happening across the campus. People can find other peers across the university with similar interests, say finding a group of people who are interested in learning Design Patterns or finding people who want to go for skiing on a particular weekend. Users can create forums to initiate discussions and peers can subscribe to the forum to participate in ongoing discussions. Users can interact with each other by posting in the forums or by replying to existing posts. Users could also unsubscribe from forums if they are no longer interested in following a group. Peers can be invited to join Campus Talk and participate in discussions.

Technologies and tools used: JavaScript & JQuery (User Interactions), Html (User Interface), Bootstrap (Responsive Design), Jersey(API Endpoints), Java (Backend), Hibernate (ORM), MySQL (Database)

Demo of the application: Click here

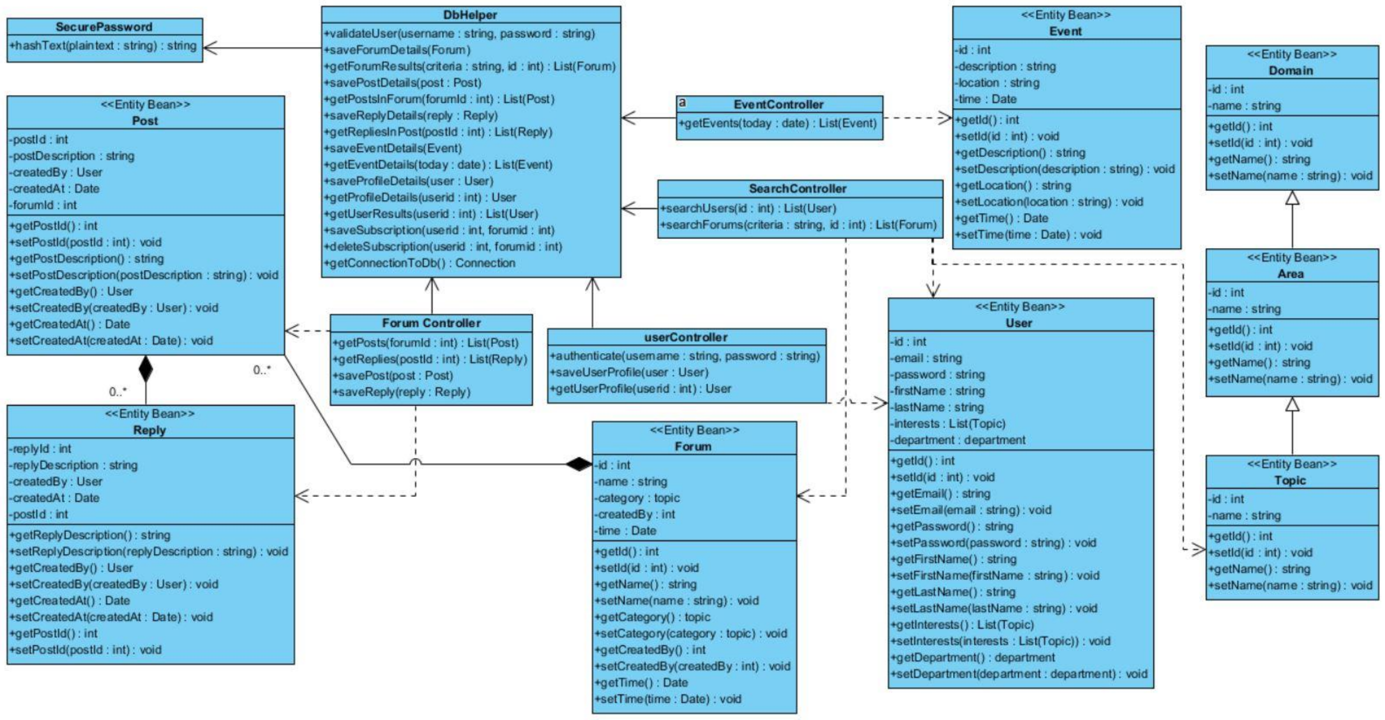
What features were implemented?

1) UR – 001: User signs up in the application
2) UR – 002: User logs in the application
3) UR – 003: User can view his/her profile
4) UR – 005: User can invite friends through email
5) UR – 006: User can view the events/posts on the news feed
6) UR – 007: User can search for forums/ people with similar interests
7) UR – 008: User can click on the displayed search results to look at a selected forum or person's profile
8) UR – 009: User can subscribe to forum
9) UR – 010: User can unsubscribe from forum
10) UR – 011: User can create a forum of his/her interest
11) UR – 012: User can create a new post in the forum.
12) UR – 013: User can reply to any post in the forum
13) UR – 014: User can logout of the application

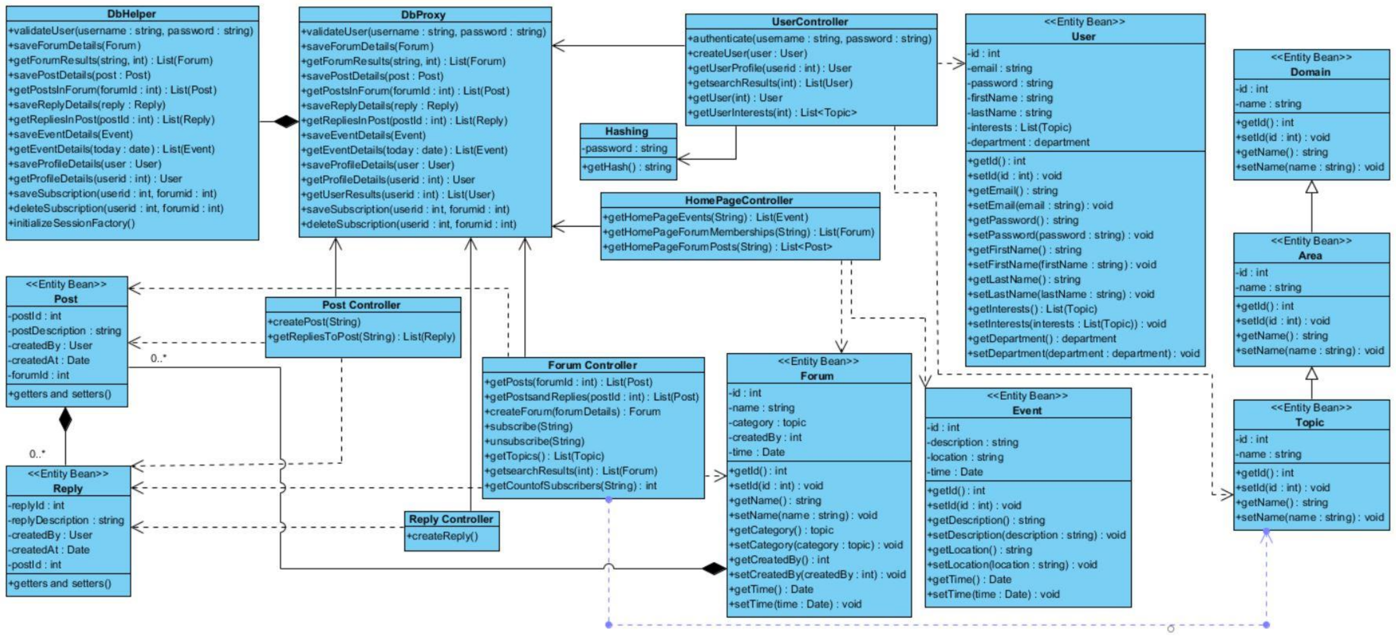
Which features were not implemented from Part 2?

1) UR – 004: User can modify his profile page

Part – 2 Class Diagram:
This class diagram can also be viewed by clicking here


SecurePassword
+hashText(plaintext : string) : string
DbHelper
+validateUser(username : string, password : string)
+saveForumDetails(Forum)
+getForumResults(criteria : string, id : int) : List(Forum)
+savePostDetails(post : Post)
+getPostsInForum(forumId : int) : List(Post)
+saveReplyDetails(reply : Reply)
+getRepliesInPost(postId : int) : List(Reply)
+saveEventDetails(Event)
+getEventDetails(today : date) : List(Event)
+saveProfileDetails(user : User)
+getProfileDetails(userid : int) : User
+getUserResults(userid : int) : List(User)
+saveSubscription(userid : int, forumid : int)
+deleteSubscription(userid : int, forumid : int)
+getConnectionToDb() : Connection
<<Entity Bean>> Event
-id : int
-description : string
-location : string
-time : Date
+getId() : int
+setId(id : int) : void
+getDescription() : string
+setDescription(description : string) : void
+getLocation() : string
+setLocation(location : string) : void
+getTime() : Date
+setTime(time : Date) : void
<<Entity Bean>> Domain
-id : int
-name : string
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
<<Entity Bean>> Post
-postId : int
-postDescription : string
-createdBy : User
-createdAt : Date
-forumId : int
+getPostId() : int
+setPostId(postId : int) : void
+getPostDescription() : string
+setPostDescription(postDescription : string) : void
+getCreatedBy() : User
+setCreatedBy(createdBy : User) : void
+getCreatedAt() : Date
+setCreatedAt(createdAt : Date) : void
EventController
+getEvents(today : date) : List(Event)
SearchController
+searchUsers(id : int) : List(User)
+searchForums(criteria : string, id : int) : List(Forum)
<<Entity Bean>> Area
-id : int
-name : string
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
Forum Controller
+getPosts(forumId : int) : List(Post)
+getReplies(postId : int) : List(Reply)
+savePost(post : Post)
+saveReply(reply : Reply)
userController
+authenticate(username : string, password : string)
+saveUserProfile(user : User)
+getUserProfile(userid : int) : User
<<Entity Bean>> User
-id : int
-email : string
-password : string
-firstName : string
-lastName : string
-interests : List(Topic)
-department : department
+getId() : int
+setId(id : int) : void
+getEmail() : string
+setEmail(email : string) : void
+getPassword() : string
+setPassword(password : string) : void
+getFirstName() : string
+setFirstName(firstName : string) : void
+getLastName() : string
+setLastName(lastName : string) : void
+getInterests() : List(Topic)
+setInterests(interests : List(Topic)) : void
+getDepartment() : department
+setDepartment(department : department) : void
0..*
0..*
<<Entity Bean>> Reply
-replyId : int
-replyDescription : string
-createdBy : User
-createdAt : Date
-postId : int
+getReplyDescription() : string
+setReplyDescription(replyDescription : string) : void
+getCreatedBy() : User
+setCreatedBy(createdBy : User) : void
+getCreatedAt() : Date
+setCreatedAt(createdAt : Date) : void
+getPostId() : int
+setPostId(postId : int) : void
<<Entity Bean>> Forum
-id : int
-name : string
-category : topic
-createdBy : int
-time : Date
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
+getCategory() : topic
+setCategory(category : topic) : void
+getCreatedBy() : int
+setCreatedBy(createdBy : int) : void
+getTime() : Date
+setTime(time : Date) : void
<<Entity Bean>> Topic
-id : int
-name : string
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void

Final Class Diagram:
This class diagram can also be viewed by clicking here


DbHelper
+validateUser(username : string, password : string)
+saveForumDetails(Forum)
+getForumResults(string, int) : List(Forum)
+savePostDetails(post : Post)
+getPostsInForum(forumId : int) : List(Post)
+saveReplyDetails(reply : Reply)
+getRepliesInPost(postId : int) : List(Reply)
+saveEventDetails(Event)
+getEventDetails(today : date) : List(Event)
+saveProfileDetails(user : User)
+getProfileDetails(userid : int) : User
+saveSubscription(userid : int, forumid : int)
+deleteSubscription(userid : int, forumid : int)
+initializeSessionFactory()
DbProxy
+validateUser(username : string, password : string)
+saveForumDetails(Forum)
+getForumResults(string, int) : List(Forum)
+savePostDetails(post : Post)
+getPostsInForum(forumId : int) : List(Post)
+saveReplyDetails(reply : Reply)
+getRepliesInPost(postId : int) : List(Reply)
+saveEventDetails(Event)
+getEventDetails(today : date) : List(Event)
+saveProfileDetails(user : User)
+getProfileDetails(userid : int) : User
+getUserResults(userid : int) : List(User)
+saveSubscription(userid : int, forumid : int)
+deleteSubscription(userid : int, forumid : int)
UserController
+authenticate(username : string, password : string)
+createUser(user : User)
+getUserProfile(userid : int) : User
+getsearchResults(int) : List(User)
+getUser(int) : User
+getUserInterests(int) : List<Topic>
<<Entity Bean>> User
-id : int
-email : string
-password : string
-firstName : string
-lastName : string
-interests : List(Topic)
-department : department
+getId() : int
+setId(id : int) : void
+getEmail() : string
+setEmail(email : string) : void
+getPassword() : string
+setPassword(password : string) : void
+getFirstName() : string
+setFirstName(firstName : string) : void
+getLastName() : string
+setLastName(lastName : string) : void
+getInterests() : List(Topic)
+setInterests(interests : List(Topic)) : void
+getDepartment() : department
+setDepartment(department : department) : void
<<Entity Bean>> Domain
-id : int
-name : string
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
Hashing
-password : string
+getHash() : string
HomePageController
+getHomePageEvents(String) : List(Event)
+getHomePageForumMemberships(String) : List(Forum)
+getHomePageForumPosts(String) : List<Post>
<<Entity Bean>> Area
-id : int
-name : string
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
<<Entity Bean>> Post
-postId : int
-postDescription : string
-createdBy : User
-createdAt : Date
-forumId : int
+getters and setters()
Post Controller
+createPost(String)
+getRepliesToPost(String) : List(Reply)
0..*
Forum Controller
+getPosts(forumId : int) : List(Post)
+getPostsandReplies(postId : int) : List(Post)
+createForum(forumDetails) : Forum
+subscribe(String)
+unsubscribe(String)
+getTopics() : List(Topic)
+getsearchResults(int) : List(Forum)
+getCountofSubscribers(String) : int
<<Entity Bean>> Forum
-id : int
-name : string
-category : topic
-createdBy : int
-time : Date
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
+getCategory() : topic
+setCategory(category : topic) : void
+getCreatedBy() : int
+setCreatedBy(createdBy : int) : void
+getTime() : Date
+setTime(time : Date) : void
<<Entity Bean>> Event
-id : int
-description : string
-location : string
-time : Date
+getId() : int
+setId(id : int) : void
+getDescription() : string
+setDescription(description : string) : void
+getLocation() : string
+setLocation(location : string) : void
+getTime() : Date
+setTime(time : Date) : void
<<Entity Bean>> Topic
-id : int
-name : string
+getId() : int
+setId(id : int) : void
+getName() : string
+setName(name : string) : void
0..*
<<Entity Bean>> Reply
-replyId : int
-replyDescription : string
-createdBy : User
-createdAt : Date
-postId : int
+getters and setters()
Reply Controller
+createReply()

The changes made in the final class diagram from Part – 2 are minimal and below is the summary:

1) Added few more controllers like PostController, ReplyController to have the functionality exist in the corresponding class controllers. This follows Single Responsibility Principle and MVC pattern.

2) To control the access to the DbHelper, we implemented the proxy design pattern by creating a proxy interface and a proxy class.

How did the design help in the development?

Doing the analysis and design of the project upfront lessened the effort to be put in implementation.

- Use case documents helped us understand each and every requirement thoroughly and realize the flow of events, errors that may arise out of user actions, validations to be handled with each use case. We were able to handle all the possible scenarios easily with this knowledge without missing any.

- Class diagram acted as a guide for the development at every stage and eased the development effort.

- Sequence and activity diagrams gave deeper insight to minute details and the interaction between classes, data flow in the entire system has become quite comprehensible

Design Patterns Used:

1) **Strategy Pattern:** We implemented Strategy pattern for the hashing function used to authenticate the user on login. Currently SHA-1 cryptographic function is being used to hash the user password before storing in the database and it is also used to authenticate the user every time he/she logs in. The implementation is done in such a way that the underlying algorithm can be changed without altering any code from client side, following Strategy design pattern.

2) **Proxy Pattern:** A proxy class is written to act as a wrapper between the client and the DbHelper class where all the scripts that interact with the database are written. This way, extra functionalities can be added or existing functionalities can be modified without changing the real code. So, there exists a flexibility to change the underlying database also without making any changes to the client code, enabling Proxy design pattern.

3) **MVC Pattern:** We have followed Model-View-Controller Architectural pattern while implementing the project to separate the concerns among view, model and controller. This allowed us to differentiate views, business rules, bean classes and the database helper classes. We were able to run the code using Driver class, even without GUIs. Editing one view updates the other views, for example when a new reply is posted in a view, instantly that reply would be reflected in the other view of posts.

Design Pattern that could have been used:

1) **Observer Pattern:** This pattern could be incorporated in the application. When a user is subscribed to a forum, all the new posts made in that forum by other users should be notified to all the subscribed users of the forum. Currently our system is not handling that functionality, new posts could be seen when the home page is refreshed, but the user is not notified.

Learning Process:

After experiencing through the whole process of software development life cycle, right from formulating the requirements, analyzing, creating various diagrams to understand the flow of the data, implementing the system and delivering it, we learnt how crucial the phase of Analysis and Design could be for any development project. Proper Analysis and Design makes the system efficient, extendible and flexible. Any new changes can be incorporated into the existing system quickly and easily.

UML Diagrams are very useful in understanding the requirements, the flow of data in the system, occurrence of the sequence of events, interaction between the classes and so on. Analyzing the design patterns that can be implemented in the code not only avoids the problems that can arise in the current system, but also helps in extending the system in future. With MVC Pattern, the concerns between various layers are separated and it reduces the coupling between classes.

We also learnt that performance bottlenecks can be avoided with accurate design of the system, for example for a given forum, getting all the posts in the forum and the replies to all the posts at one shot is faster and more efficient than making a database call for every post to get the corresponding replies. In addition to all this, documents and diagrams made as part of the analysis and design phase are self explanatory and helps any one to understand the system easily.