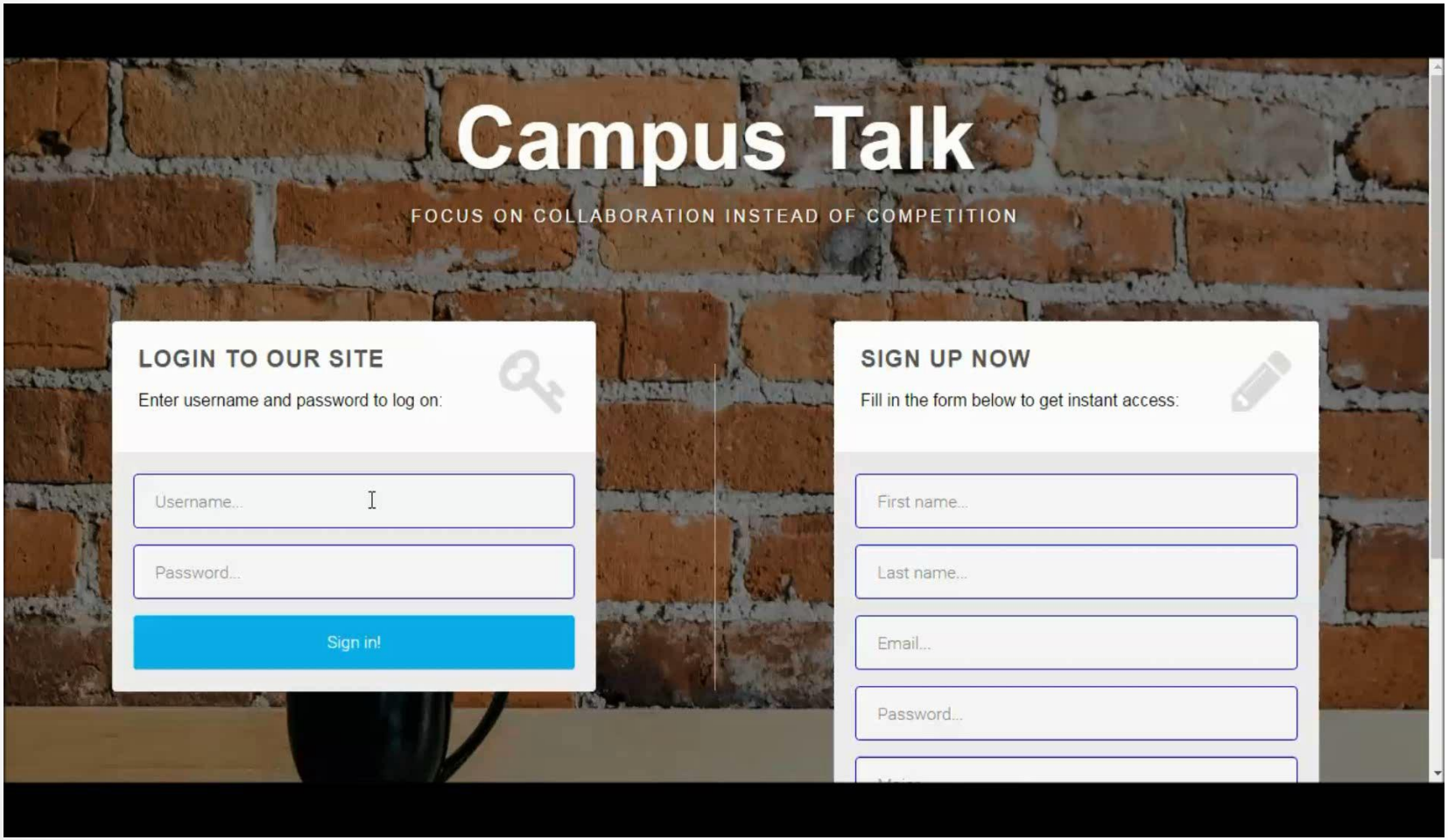
Campus Talk

Aadish Gupta (CSCI 5448), Pallavi Madasu(CSCI 5448),
Yogitha Mahadasu(CSCI 5448)

Product Demo

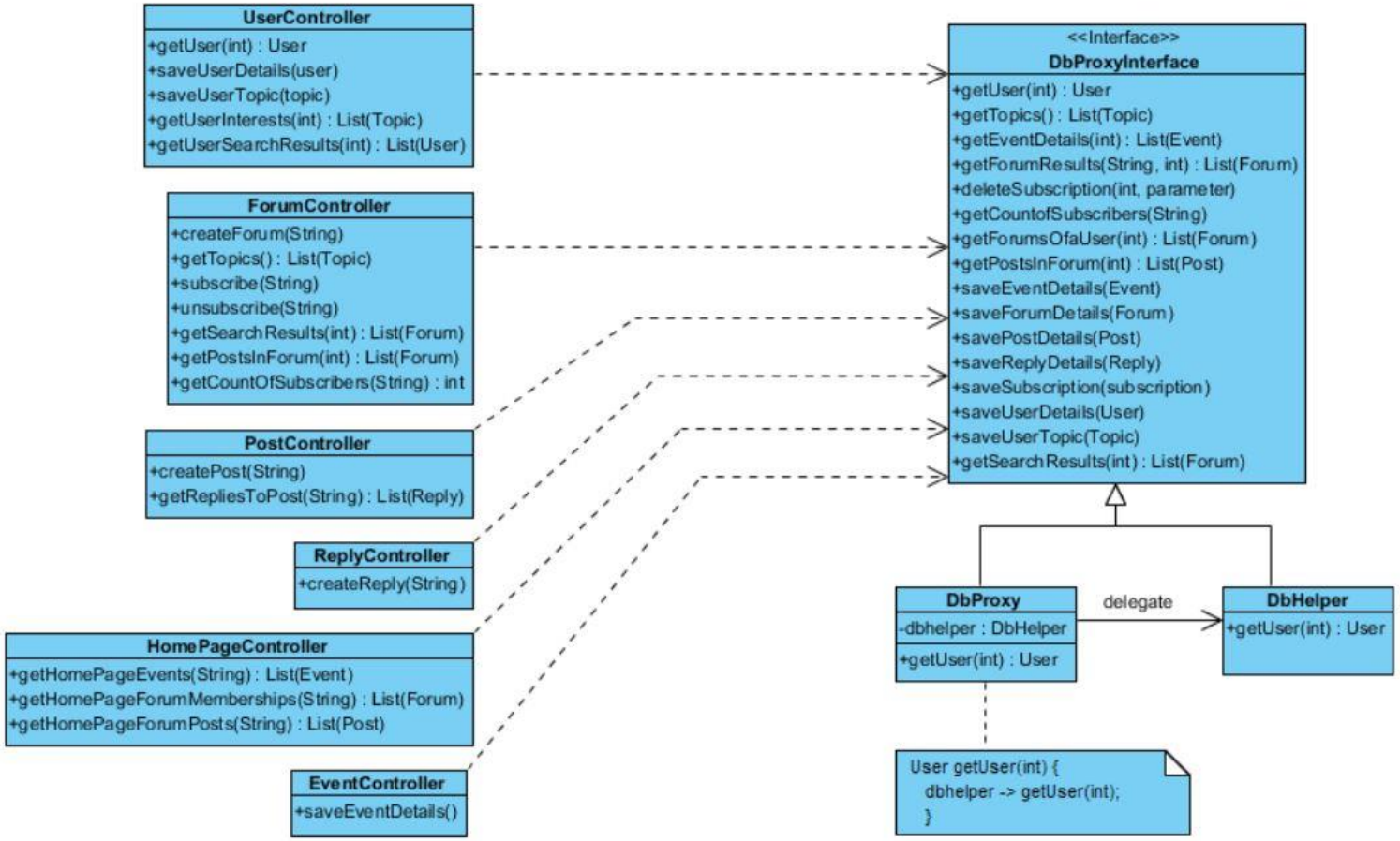


https://github.com/Yogitha6/Campus-Talk/blob/master/CampusTalk_Video.mp4

Design Patterns

- Proxy Design Pattern
- Strategy Design Pattern

Proxy Design Pattern



Strategy Design Pattern

- To store the user password in database we employ hashing
- Hashing can be performed using various algorithms
- Aim of using this design pattern
  - Make our system resilient to changes
  - No changes in business logic

Code for Hashing Class

```
public class Hashing {

    private String password;

    public Hashing(String password){
        this.password = password;
    }

    public String getHash(){
        return DigestUtils.sha1Hex(password);
    }
}
```

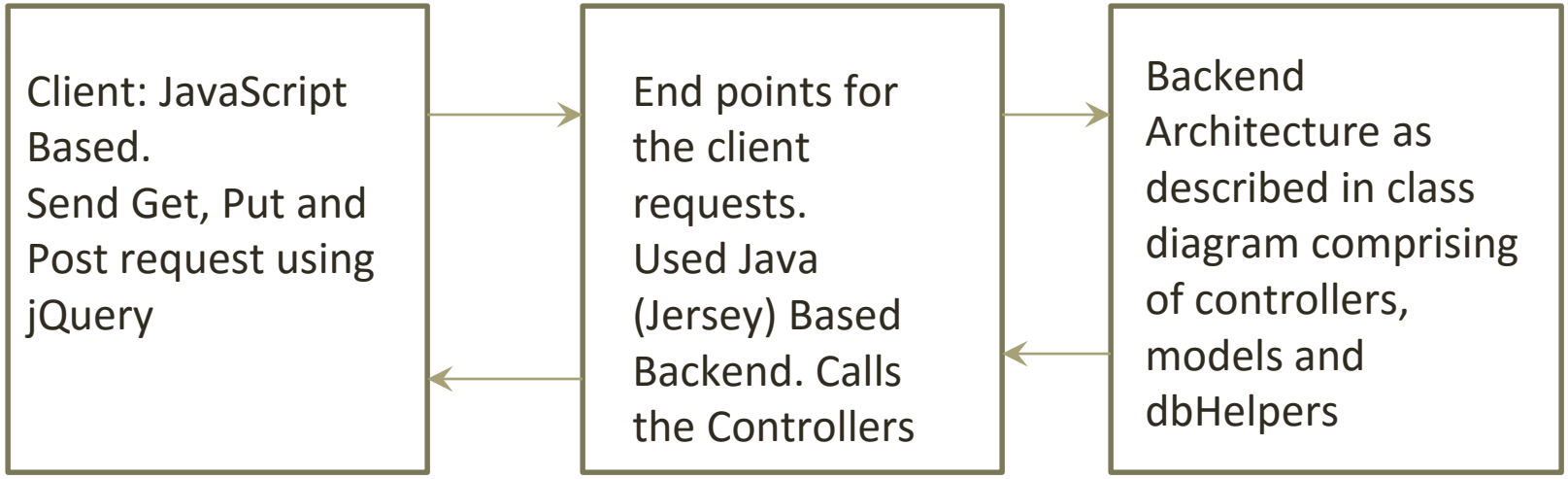
Use of Hashing Class in Business Logic (Login)

```
int userId=0;
Hashing hash = null;
try {
    JSONObject json = new JSONObject(userCredentials);
    String username = json.getString("name");
    String password = json.getString("pwd");
    hash = new Hashing( password );
    String hashedPassword = hash.getHash();
    User user = dbproxy.getUser(username);
    String actualPassword = user.getPassword();
```

What we want to share with the class

- World is moving towards API's
- Use MVC architectural pattern
- It is not so complex as it looks
- Makes the flow of system easy to understand
- Don't require you to be jack of all trades
- We implemented our backend in Java
- Jersey is a great light weighted easy to use library to create API's in Java

Our System Architecture in Layman Representation



Thanks Folks!!!

- It was a great learning experience !!!
- We had loads of fun while building this project!!!