**INTRODUCTION**

* 1. **Introduction:**

The goal is to compile all event and activity information in one easily accessible location. The application is constructed using the academic techniques we acquired while working on the project. It is a web-hosted program with database administration capabilities and a stunning, user-friendly interface that was created by drawing design cues from top-tier websites.

We are utilizing the following technologies: ORM, Servlet, Eclipse IDE, and MySQL.

**1.2 Problem-Statement:**

*“College News Report”*

**1.3 Scope of Research:**

The project required expertise in using Eclipse IDE which is the tool preferred for developing the idea we formulated from the problem statement. The application requires the usage of ORM. We used hibernate to achieve this. The advantage of using an ORM is that we are database independent which lets us use any database. Servlets are also used to communicate with the web interface and perform the actions that we desire.

**1.4 Objectives:**

* Application to display all the News and events
* Create storage system using ORM and MYSQL
* Admin functionality
  + Add articles
  + Edit and delete existing article
* Design User-friendly website
* Create servlets to achieve the listed functionality
* Host the application on Apache Tomcat

**LITERATURE REVIEW**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Authors** | **Title** | **Publishing year** |
| 1 | O'Neil, Elizabeth J | Object/relational mapping 2008: hibernate and the entity data model (edm) | 2008 |
| 2 | Perry, Bruce W | Java servlet & JSP cookbook. | 2015 |
| 3 | Burnette, E. | Eclipse IDE Pocket Guide | 2010 |
| 4 | Hemrajani, Anil | Agile Java Development with Spring, Hibernate and Eclipse | 2010 |

**HARDWARE & SOFTWARE REQUIREMENTS**

**3.1 Hardware Requirements:**

Processor:

* Intel: i3 5th generation and above
* Apple: M1 and above
* AMD: Ryzen 4000 series and above

Ram: 8GB and above

Storage: 64GB and above

**3.2 Software Requirements:**

* Eclipse IDE
* MYSQL
* MYSQL Connector driver
* Chrome, Edge or Safari

**FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS**

**4.1 Functional Requirements:**

* Account creation
* Login
* View news articles
* Admin account functionality
* Editing functionality
* Deletion functionality

**4.2 Nonfunctional Requirements:**

* ORM implementation [2]
* Webhosting in Apache server [3]
* Store pictures of the articles [4]

**IMPLEMENTATION**

**5.1 WORKFLOW**

**Diagram

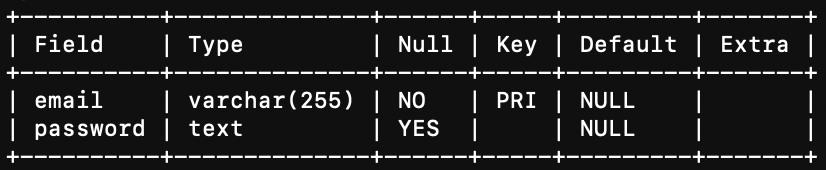
Description automatically generated**

**5.2 MYSQL SETUP**

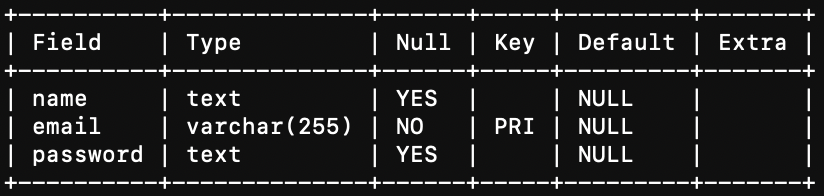
**Text

Description automatically generated with low confidence**

(Necessary Tables)



(Admin Table)



(User table)

A picture containing table

Description automatically generated

(News article table)

**5.3 HTML/JSP Files and functionality**

* Login.html (User login Page)
* AdminIndex.html (admin control center)
* Delete.jsp (Article delete page for admin)
* DeleteNews.jsp (Article delete functionality)
* Display.jsp (Displays articles)
* Edit.jsp (article edit page for admin)
* EditNews.jsp (article edit functionality for admin)
* EnterNews.jsp (article entry page for admin)
* NewsChannel.jsp (list of articles)
* Register.jsp (User registry page)

**5.4 Servlet Files and functionality**

* InsertPeople.java (Registers new users)

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

doGet(request, response);

String name = request.getParameter("name");

String email = request.getParameter("email");

String pass = request.getParameter("password");

people p = **new** people();

p.setEmail(email);

p.setName(name);

p.setPassword(pass);

peopledao dao = **new** peopledao();

**if** (dao.addBook(p) == 0 )

response.sendRedirect("login.html");

**else**

response.sendRedirect("register.jsp");

}

* People.java (Model class for Hibernate)

**package** insertServ;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="people")

**public** **class** people {

@Id

@Column(name="name", updatable=**false**)

**private** String name;

@Column(name = "email", updatable=**false**)

**private** String email;

@Column(name = "password", updatable=**false**)

**private** String password;

**public** people (String name, String email, String password) {

**super**();

**this**.name = name;

**this**.email = email;

**this**.password = password;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** people() {

// **TODO** Auto-generated constructor stub

}

}

* Peopledao.java (Hibernate functionality)

**package** insertServ;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**public** **class** peopledao {

**public** **int** addBook(people book) {

**try** {

SessionFactory sessionFactory = peopleutil.*getSessionFactory*();

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

session.saveOrUpdate(book);

transaction.commit();

session.close();

}

**catch**(Exception e) {

**return** 1;

}

**return** 0;

}

}

* Peopleutil.java (Configuration function for hibernate)

**package** insertServ;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**public** **class** peopledao {

**public** **int** addBook(people book) {

**try** {

SessionFactory sessionFactory = peopleutil.*getSessionFactory*();

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

session.saveOrUpdate(book);

transaction.commit();

session.close();

}

**catch**(Exception e) {

**return** 1;

}

**return** 0;

}

}

* loginPeople.java (login servlet)

PrintWriter out = response.getWriter();

ResultSet rs;

ResultSet adminrs;

String userid = request.getParameter("email");

String password = request.getParameter("password");

**try**{

String sql = "SELECT \* FROM people WHERE email = ? AND password = ?";

String adminsql = "select \* from admin where email = ? and password = ?";

Class.*forName*("com.mysql.jdbc.Driver");

Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/news", "root", "tester123");

PreparedStatement preparedStatement = con.prepareStatement(sql);

preparedStatement.setString(1, userid);

preparedStatement.setString(2, password);

PreparedStatement preparedStatementa = con.prepareStatement(adminsql);

preparedStatementa.setString(1, userid);

preparedStatementa.setString(2, password);

rs =preparedStatement.executeQuery();

adminrs = preparedStatementa.executeQuery();

**if**(adminrs.next()) {

response.sendRedirect("AdminIndex.html");

}

**else**

{

**if**(rs.next()) {

response.sendRedirect("NewsChannel.jsp");

}

**else**

response.sendRedirect("login.html");

}

}

**catch**(Exception e) {

System.***out***.println("The exception is" + e);

}

**RESULTS**

**Graphical user interface, application, website

Description automatically generated**

(Register page)

Graphical user interface, application, website

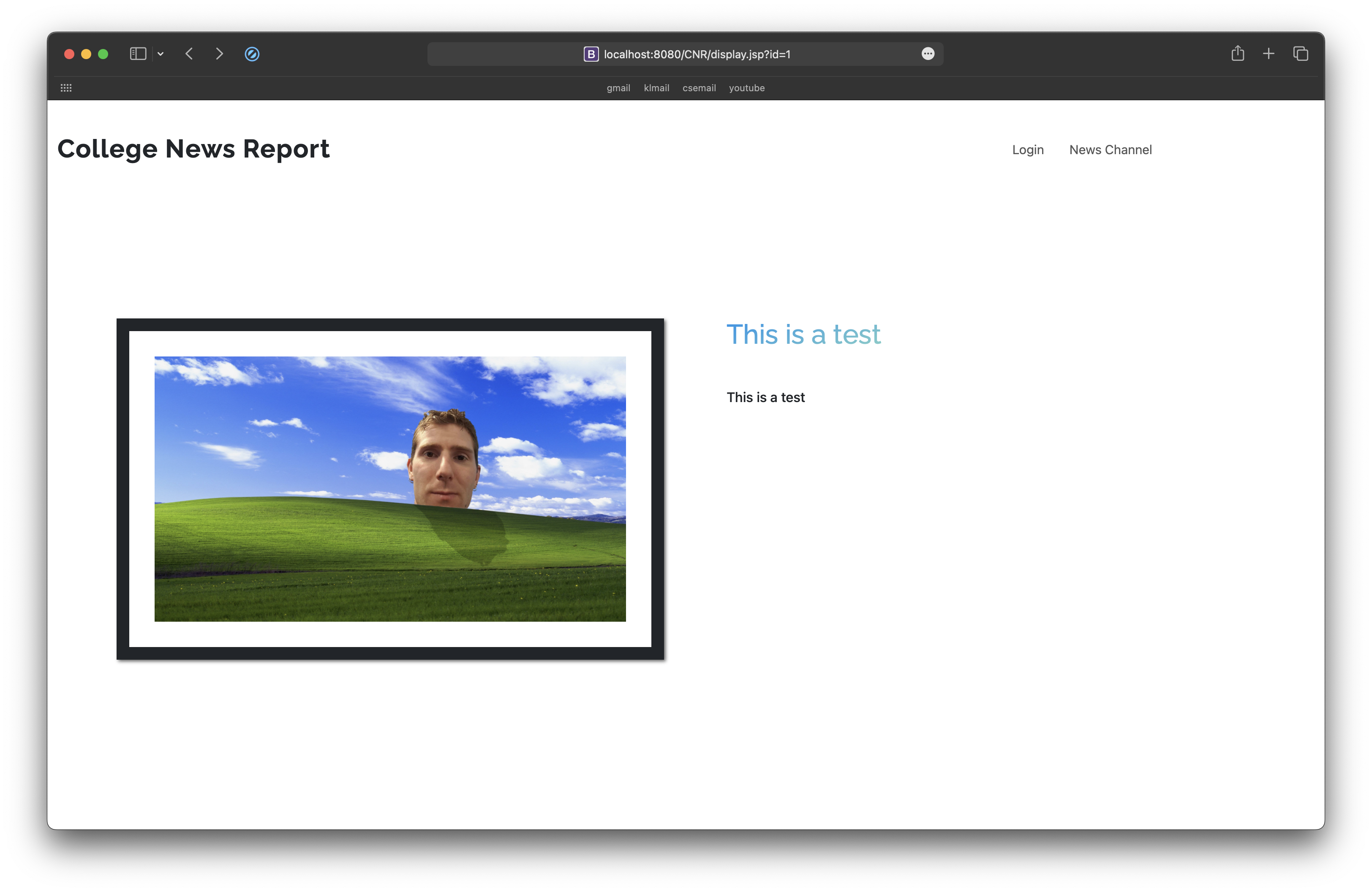
Description automatically generated

(Login Page)

Graphical user interface, application

Description automatically generated

(News Channel)

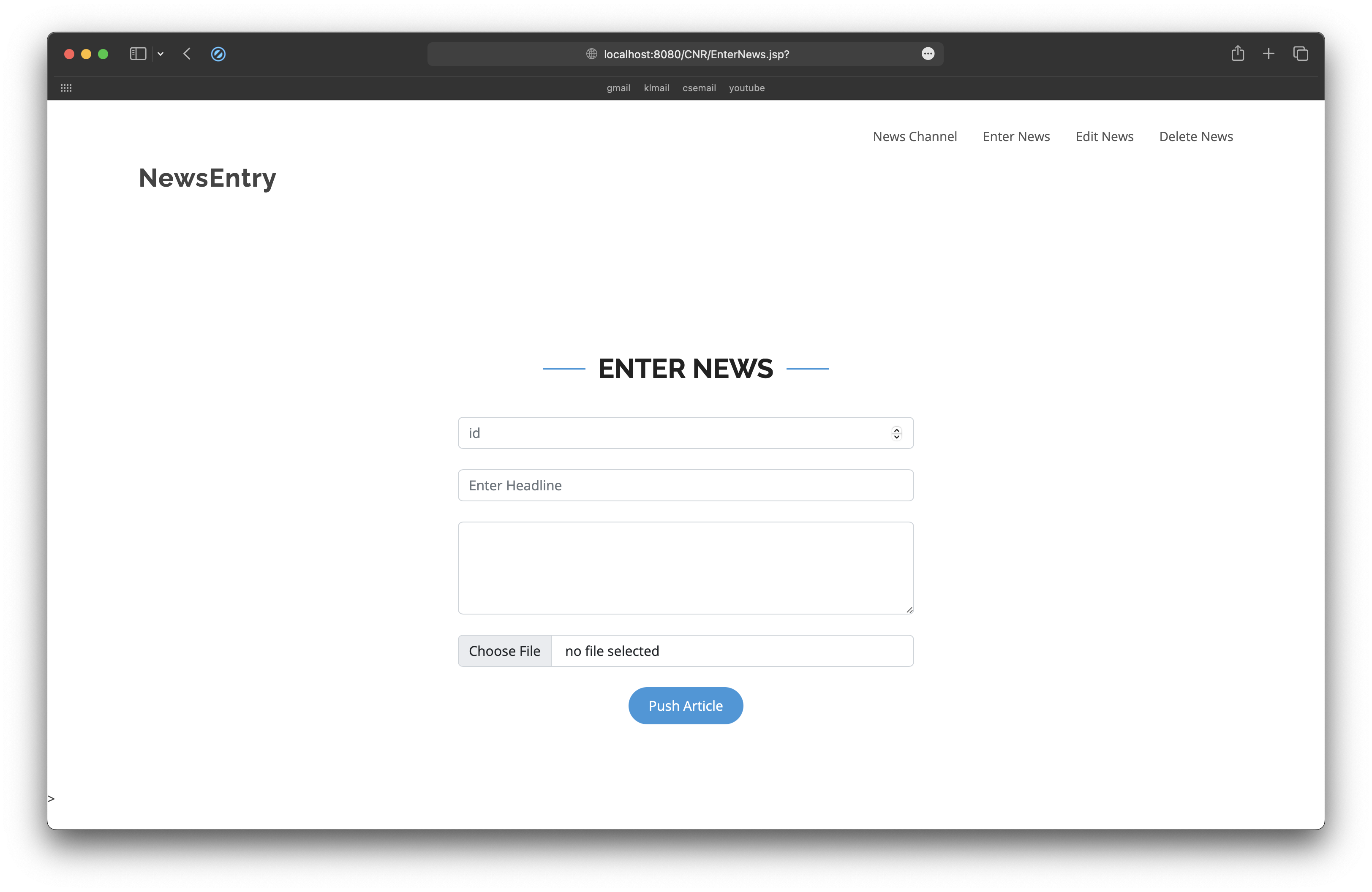


(news article)

Graphical user interface, text, application

Description automatically generated

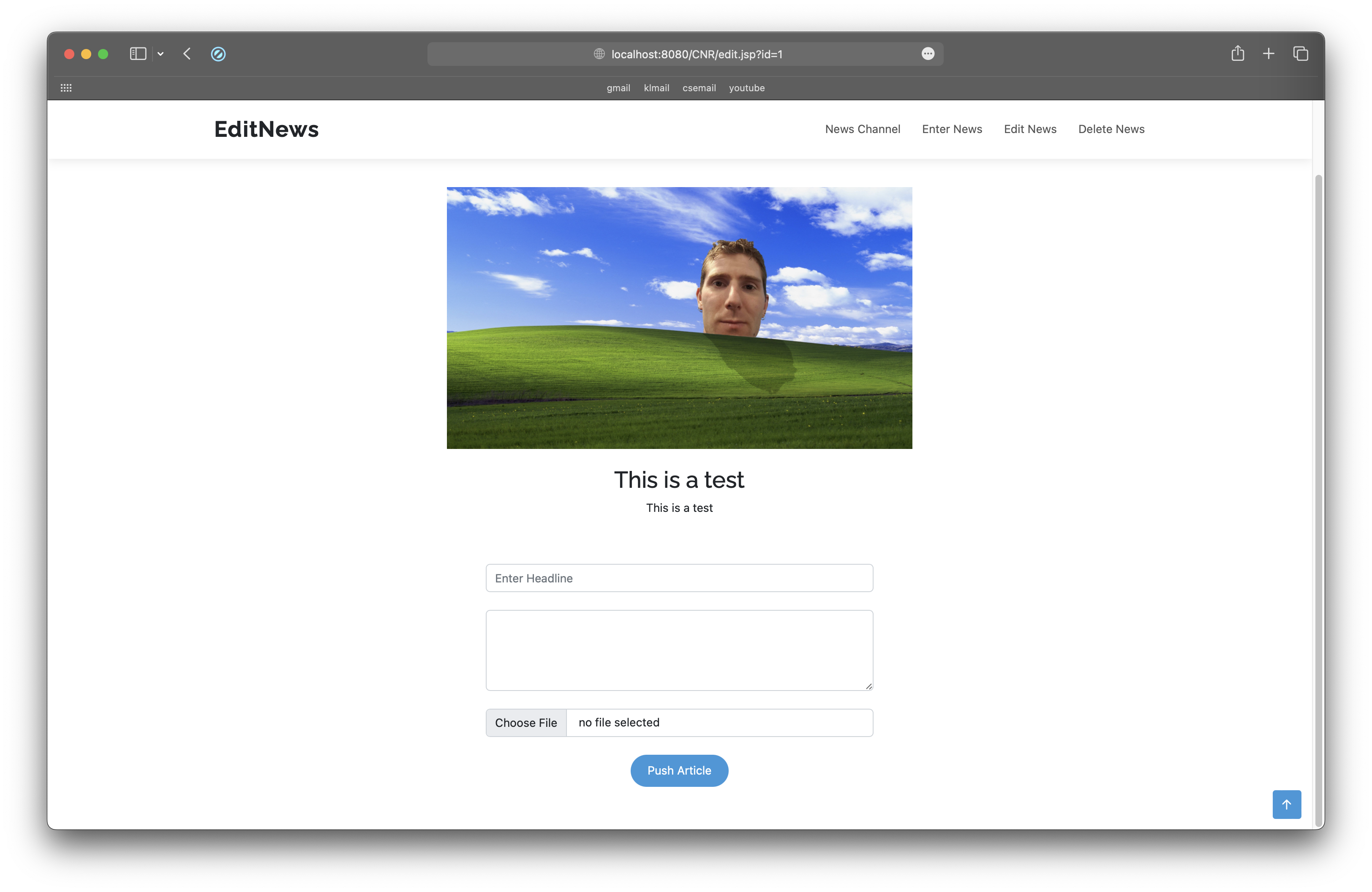
(admin control center)



(news entry page)



(edit news select page)



(editing page)

Graphical user interface, text, application

Description automatically generated

(delete news select page)

**CONCLUSION**

To sum up, the application seeks to address the issue that it can be challenging to keep track of all the enjoyable activities and events that take place at each college. To address this issue that we all face, the plan is to gather all this data about occasions and activities in a single, openly accessible area. The academic methods we learned while working on the project served as the foundation for this application. It is a web-hosted application with database administration features and a gorgeous, approachable interface that was developed by taking design cues from top websites.

We are using the ORM, Servlet, Eclipse IDE, and MySQL technologies.

A relatively lightweight application that can be hosted anywhere and customised to suit the system requirements is produced as a consequence of the use of these technologies.[1]

**REFERENCES**

|  |  |
| --- | --- |
| <https://quillbot.com> | [1] |
| <https://hibernate.org> | [2] |
| <https://www.eclipse.org/webtools/jst/components/ws/1.5/tutorials/InstallTomcat/InstallTomcat.html> | [3] |
| <https://www.geeksforgeeks.org/how-to-add-image-to-mysql-database-using-servlet-and-jdbc/> | [4] |