

User Management in Authentication of Users in BISU-Bilar Campus Library Management System

JOMAR TRESMONTE JUSTOL (Student)

Bachelor of Science in Computer Science 3, College of Technology, and Allied Sciences, Bohol Island State University-Bilar Campus, jomar.justol@bisu.edu.ph

MAX ANGELO DAPITILLA PERIN (INSTRUCTOR)

Faculty of Department of Computer Science, College of Technology and Allied Sciences, Bohol Island State University-Bilar Campus, maxangelo.perin@bisu.edu.ph

Authentication of users in the user management of a library management system ensures that only authorized individuals can access the system, promoting security and privacy of user information and resources. The library management system employs various authentication methods, such as username/password combinations or biometric authentication, to verify the identity of users during the login process, ensuring secure access. It focuses by User Management in Authentication of Users in database, design as full stack web development. Additionally, the system may implement multi-factor authentication, requiring users to provide additional credentials or undergo additional verification steps to enhance the overall security of the system.

CCS CONCEPTS • Software and its Engineering • Software creation and management • Designing software

Additional Keywords and Phrases: authentication, management, security

ACM Reference Format:

Jomar Tresmonte Justol, MAX ANGELO DAPITILLA PERIN. 2023. **User Management in Authentication of Users in BISU-Bilar Campus Library Management System**. In Research Project Presentation for Bachelor of Science in Computer Science 3 in CS 324 – Web Development and Enterprises S.Y. 2022-2023, 2nd Semester, Bohol Island State University-Bilar Campus, Zamora, Bilar, Republic of the Philippines. ACM, New York, NY, USA

1 INTRODUCTION

User management in a Library Management System is a critical component that facilitates the efficient administration of library resources and services. It involves the management of user accounts, privileges, and access levels within the system to ensure smooth operations and effective utilization of library resources. User management functionalities typically include user registration, authentication, and authorization processes, allowing library staff and patrons to access various features such as borrowing and returning books, accessing digital resources, and managing personal profiles. Through user management, librarians can effectively track user activities, maintain borrowing records, implement fine and penalty systems, and tailor services based on individual needs. By implementing robust user management practices, Library Management Systems can streamline operations, improve user experience, and ensure the proper utilization of library resources.

This article describes the development and application of a structural equation model which allows librarians to quantitatively measure library users' perceived quality, satisfaction and loyalty with a library as well as the degree to which specific elements of a library's services, collections and environment contribute to those perceptions. The article reports the results of a survey among users at five Danish libraries with particular attention to the Copenhagen Business [School Library](#) [1].

This study investigated the factors that influence behavioral intention to use a digital library system in different developing countries through the TAM framework developed by Davis (1986). The findings generally supported the hypotheses derived from the model as well as earlier empirical studies [2].

2 PROPOSED METHODOLOGY

In this section, the system design, and the importance for the application of User Management in Authentication of Users in BISU-Bilar Campus Library Management System will be thoroughly discussed.

2.1 Database

The [Figure 1](#) is the “bisubilarlms” database that consists of a single table called “user_add” with five columns. This columns contains the data and information provided by the user.

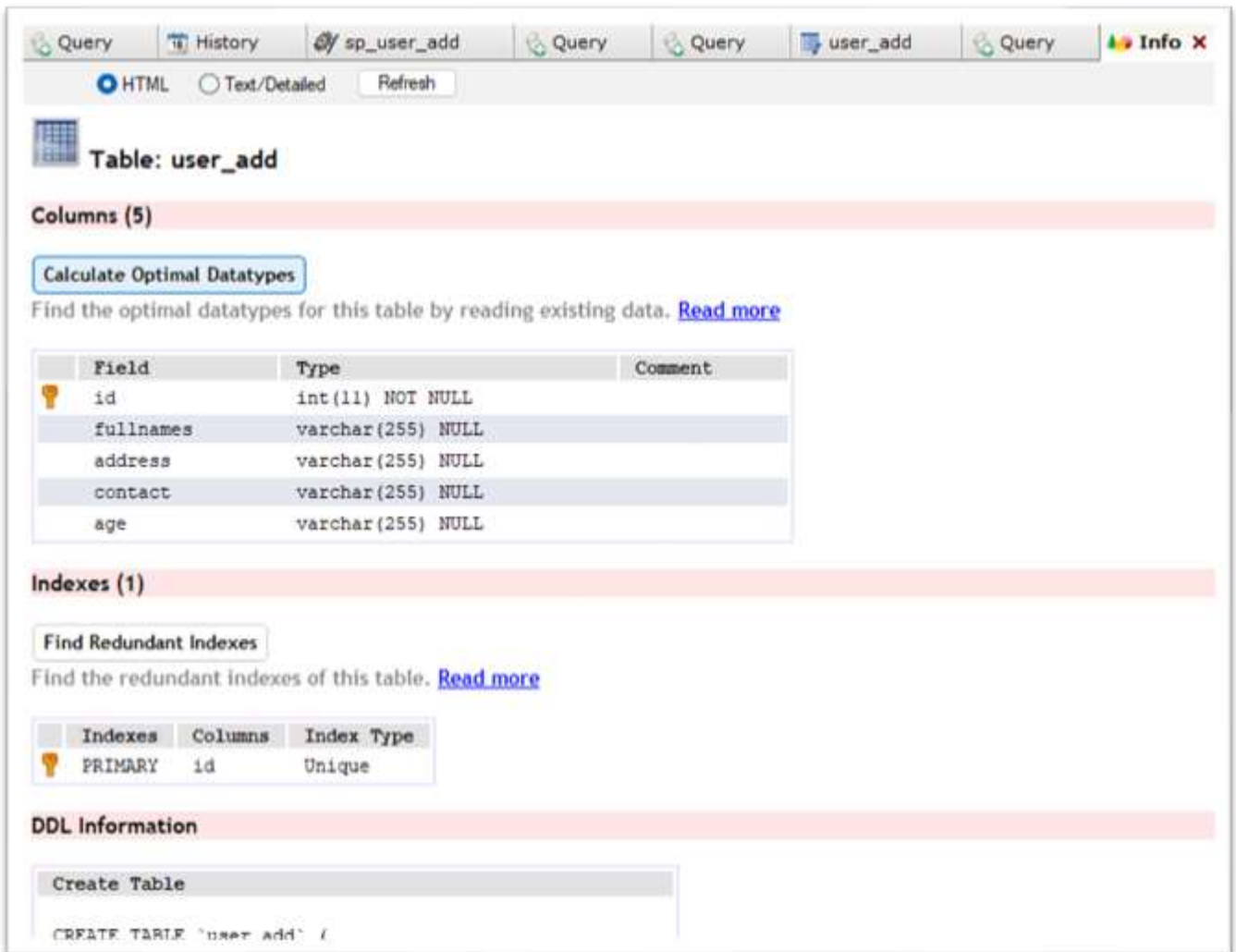


Table: user_add

Columns (5)

Calculate Optimal Datatypes

Find the optimal datatypes for this table by reading existing data. [Read more](#)

Field	Type	Comment
id	int(11) NOT NULL	
fullnames	varchar(255) NULL	
address	varchar(255) NULL	
contact	varchar(255) NULL	
age	varchar(255) NULL	

Indexes (1)

Find Redundant Indexes

Find the redundant indexes of this table. [Read more](#)

Indexes	Columns	Index Type
PRIMARY	id	Unique

DDL Information

Create Table

```
CREATE TABLE `user_add` (
```

Figure 1: Table and columns of bisubilarlms database

```
DELIMITER $$
```

```
USE `users`$
```

```

DROP PROCEDURE IF EXISTS `sp_user_add`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_user_add`(
    IN _fullnames VARCHAR(255),
    IN _age VARCHAR(255),
    IN _address VARCHAR(255),
    IN _contact VARCHAR(255)
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        BEGIN
            ROLLBACK;
            RESIGNAL;
        END;
    START TRANSACTION;
    -- #####START#####
    SET @_users := (SELECT COUNT(a.fullnames) FROM user_add a WHERE a.fullnames = _fullnames);
    IF @_users = 0
    THEN
        SET @_users := (SELECT COUNT(a.address) FROM user_add a WHERE a.address = _address);
        IF @_users = 0
        THEN
            INSERT INTO user_add (fullnames, address, contact, age)
            VALUES (_fullnames, _address, _contact, _age);
            SELECT 'users_add_successfully' _ret;
        ELSE
            SELECT 'duplicate_entry' _ret;
        END IF;
    ELSE
        SELECT 'duplicate_entry' _ret;
    END IF;
    -- #####END#####
    COMMIT;
    END$$
DELIMITER ;

```

2.2 Server/API

In [Figure 2](#), the localhost:5001/occupation/save had been tested using post method to validate if the API is connected to the database and whether it can add the input data of the user.

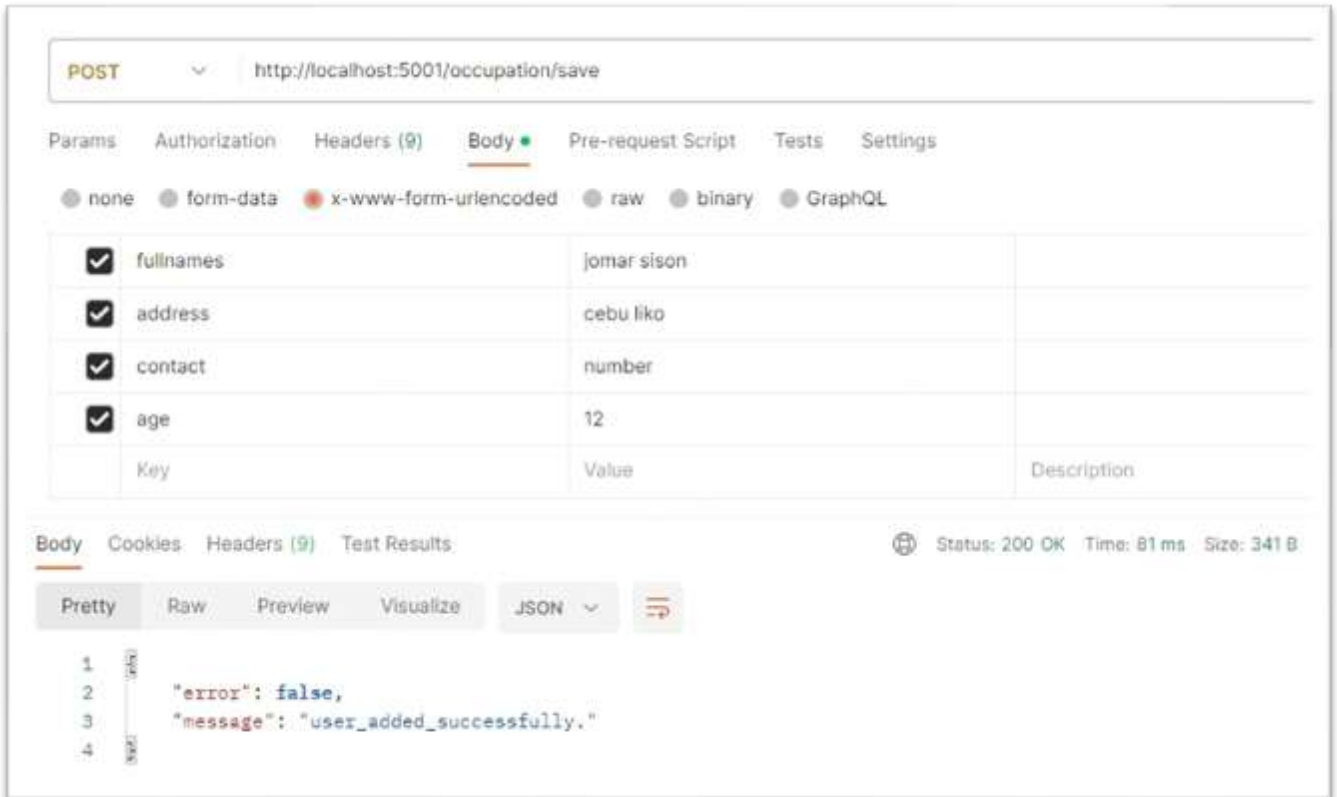


Figure 2: Testing API by adding user information.

```
occupation.post('/save', (req, res) => {  
  db.sequelize.query("CALL sp_user_add(:fullnames, :address, :contact, :age)", {  
    replacements: {  
      fullnames: req.body.fullnames,  
      address: req.body.address,  
      contact: req.body.contact,  
      age: req.body.age,  
    }  
  }).then(data => {  
    ret = data[0]["_ret"];  
  })  
})
```

```

    if (ret === "add_successfully") {
        res.send({error: false, message: 'user_add_successfully'});
    }
    else if (ret === "edit_successfully") {
        res.send({error: false, message: 'user_updated_successfully'});
    } else if (ret === "invalid_user_duplicate") {
        res.send({error: false, message: 'invalid_user_duplicate'});
    }
    else {
        res.send({error: false, message: 'user_added_successfully.'});
    }
}).catch(err => {
    res.send({ error: true, message: `Error 767: ${err}` });
});
});

```

REFERENCES

<bib id="bib1"><number>[1]</number>Martensen, A., & Grønholdt, L. (2003). Improving library users' perceived quality, satisfaction and loyalty: an integrated measurement and management system. *The Journal of Academic Librarianship*, 29(3), 140-147.</bib>
 <bib id="bib2"><number>[2]</number>Park, N., Roman, R., Lee, S., & Chung, J. E. (2009). User acceptance of a digital library system in developing countries: An application of the Technology Acceptance Model. *International journal of information management*, 29(3), 196-209.</bib>

APPENDICES

Visual Studio Code IDE

```
template.routes.js  profile.routes.js  admins.routes.js  X
routes > AS admins.routes.js > admins.post('/register') callback > replacements
1  const express = require("express");
2  const admins = express.Router();
3  //const cors = require("cors");
4  const db = require("../database/config");
5  //const csrf = require('csrf');
6  //const config = require('../database/config.json');
7
8  //const security = require('../database/security');
9
10 admins.post('/register', (req, res) => {
11     db.sequelize.query("CALL sp_user_add(:name, :student_id, :department, :year_lvl, :address, :username, :password)",
12         replacements: {
13             name: req.body.name,
14             student_id: req.body.student_id,
15             department: req.body.department,
16             year_lvl: req.body.year_lvl,
17             address: req.body.address,
18             username: req.body.username,
19             password: req.body.password,
20         }
21     ).then(data => {
22         ret = data[0]["_ret"];
23         if (ret === "add_successfully") {
24             res.send({error: false, message: 'user_add_successfully'});
25         }
26         else if (ret === "edit_successfully") {
27             res.send({error: false, message: 'user_add_successfully'});
28         }
29         else if (ret === "invalid_username_name_duplicate") {
30             res.send({error: false, message: 'duplicate_entry'});
31         }
32         else {
```

```
const express = require("express");
const occupation = express.Router();
//const cors = require("cors");
const db = require("../database/config");
//const csrf = require('csurf');
//const config = require("../database/config.json");
//const security = require("../database/security");

occupation.post('/save', (req, res) => {
  db.sequelize.query("CALL sp_user_add(:fullnames, :address, :contact, :age)", {
    replacements: {
      fullnames: req.body.fullnames,
      address: req.body.address,
      contact: req.body.contact,
      age: req.body.age,
    },
  }).then(data => {
    //res.json(data);
  });
});

const server = app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Github Contribution

<https://github.com/MAKASA-LABORATORY/186-AOUBBCLMS/tree/dev>

MAKASA-LABORATORY / 186-AOUBBCLMS · Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

main

Commits on Jun 5, 2023

- update
AmmarDSIT committed 3 minutes ago
42a7f24
- Update XOR6
AmmarDSIT committed 51 minutes ago
ca1886

Commits on Jun 2, 2023

- update documentation and database
AmmarDSIT committed 3 days ago
09a073e

Commits on May 31, 2023