```swift
//
//  CreateAccountViewController.swift
//  AttendanceApplication
//

import UIKit
import Alamofire

class CreateAccountViewController: UIViewController {

    // MARK: IBOutlets (Text Fields)

    @IBOutlet weak var name: UITextField!
    @IBOutlet weak var username: UITextField!
    @IBOutlet weak var password: UITextField!
    @IBOutlet weak var confPassword: UITextField!

    // MARK: IBOutlets (Error Labels)

    @IBOutlet weak var emailInUseLabel: UILabel!
    @IBOutlet weak var passwordsMatchLabel: UILabel!
    @IBOutlet weak var connectionErrorLabel: UILabel!
    @IBOutlet weak var signInLabel: UILabel!

    // MARK: UIView Methods

    override func viewWillAppear(_ animated: Bool) {
        resetLabels()
    }

    // MARK: IBAction Methods

    @IBAction func createAccountButton(_ sender: Any) {
        if name.text != "" && username.text != "" && password.text != "" &&
         confPassword.text != "" {
           createAccount()
        }
    }

    @IBAction func cancelButton(_ sender: Any) {
        // Button to exit create account view --> goes to login view.
        dismiss(animated: true, completion: nil)
    }

    // MARK: Internal Methods

    internal func createAccount() {
        let parameters: Parameters = [
            "type": "universal.create_account",
            "args": [
                "name": name.text,
                "username": username.text,
                "password": password.text,
                "conf_password":confPassword.text,
```

```
    ]
]
Alamofire.request(HTTPHelper.url, method: .post, parameters: parameters,
 encoding: JSONEncoding.default).responseJSON {
    response in

    switch response.result {
    case .failure( _):
        // Unable to send request; hence, connection error.
        self.setConnectionError()
        return

    case .success(let data):
        if let headerFields = response.response?.allHeaderFields as?
         [String: String], let URL = response.request?.url {
            // Create the cookies object
            let cookies = HTTPCookie.cookies(withResponseHeaderFields:
             headerFields, for: URL)
            // Add the cookies object to the httpCookieStorage

             Alamofire.SessionManager.default.session.configuration.httpCook
             ieStorage?.setCookies(cookies, for: URL, mainDocumentURL: nil)
        }

        // First make sure you got back a dictionary if that's what you
         expect
        guard let json = data as? [String : AnyObject] else {
            print("Failed to get expected response from webserver.")
            return
        }

        // Make sure this is the actual key/value types that are expected
        guard let success = json["successful"] as? Int, let reason =
         json["reason"] as? String else {
            print("Failed to get data from webserver")
            return
        }

        if success == 1 {
            // Successful creation of account
            self.resetLabels()
            self.signInLabel.isHidden = false
        } else if reason == "pass_match" {
            // Passwords do not match error
            self.setPasswordsDoNotMatch()
        } else if reason == "email_use" {
            // Email already in use error
            self.setEmailinUse()
        } else {
            // Connection error
            self.setConnectionError()
        }
    }
}
```

```
    }

    internal func setPasswordsDoNotMatch() {
        // Function for displaying the passwords do not match error
        passwordsMatchLabel.isHidden = false
        password.layer.borderColor = UIColor.red.cgColor
        password.layer.borderWidth = 1
        confPassword.layer.borderColor = UIColor.red.cgColor
        confPassword.layer.borderWidth = 1
    }

    internal func setEmailinUse() {
        // Function for displaying the email in use error
        emailInUseLabel.isHidden = false
        username.layer.borderColor = UIColor.red.cgColor
        username.layer.borderWidth = 1
    }

    internal func setConnectionError() {
        // Function for displaying the connection error
        connectionErrorLabel.isHidden = false
    }

    internal func resetLabels() {
        // Resets all errors and successes, called when view is opened
        name.text = ""
        username.text = ""
        password.text = ""
        confPassword.text = ""
        emailInUseLabel.isHidden = true
        passwordsMatchLabel.isHidden = true
        connectionErrorLabel.isHidden = true
        signInLabel.isHidden = true
        password.layer.borderWidth = 0
        confPassword.layer.borderWidth = 0
        username.layer.borderWidth = 0
    }
}
```