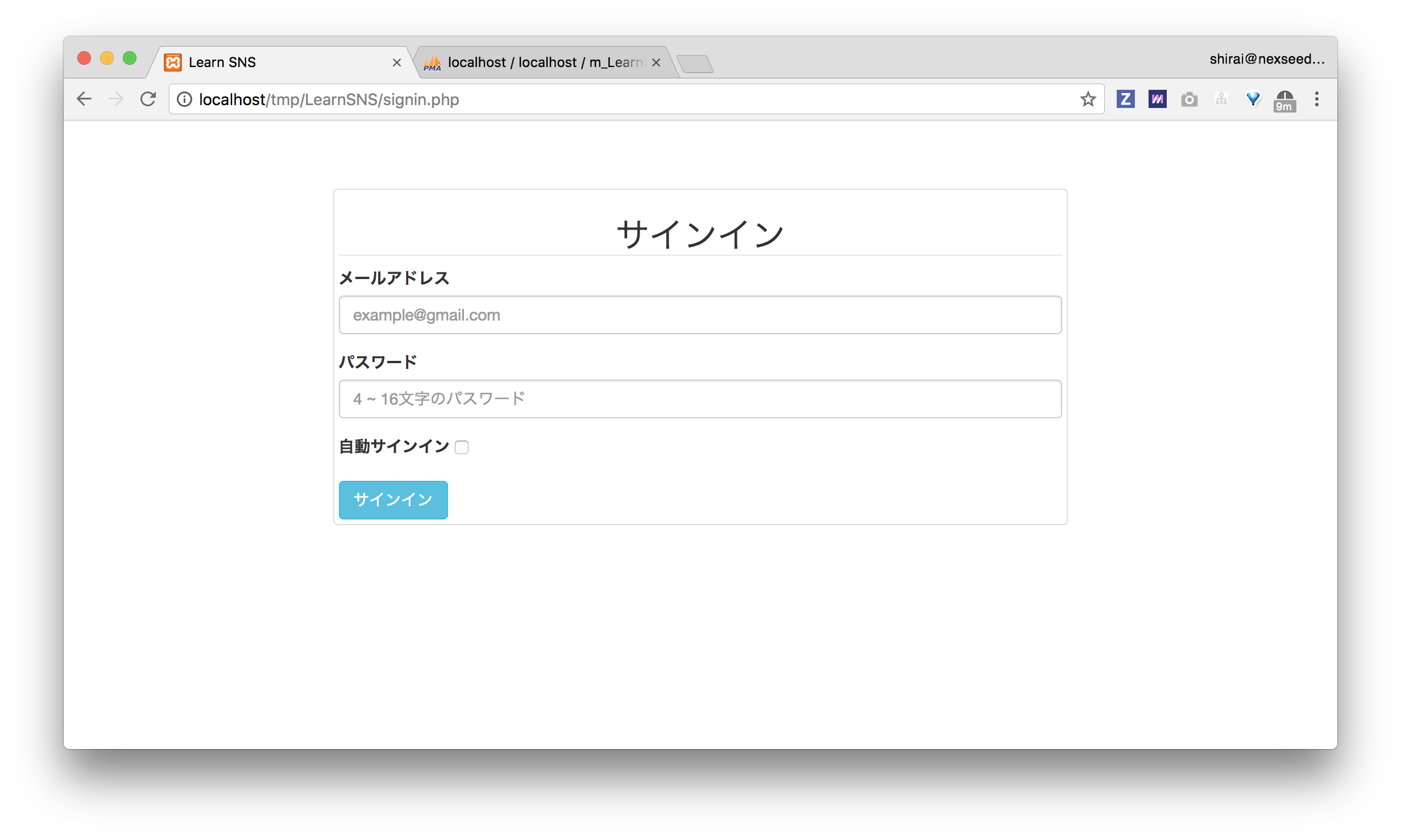
# **SNSシステム開発 - サインイン機能**

ユーザー登録機能が完成したので、ここからは**サインイン機能**の開発に入っていきます。

## **概要**

このページでは、サインイン機能の作成を題材に下記について学習していきます。

1. サインインフォームのバリデーション
2. サインイン処理
   * データベースに登録されたデータとの照合
   * password\_verify()関数を使ったハッシュ文字列との比較
   * $\_SESSIONへサインイン情報の保存

メールアドレスとパスワードを正しく入力した時と、そうでない時で条件分岐する部分が重要です。

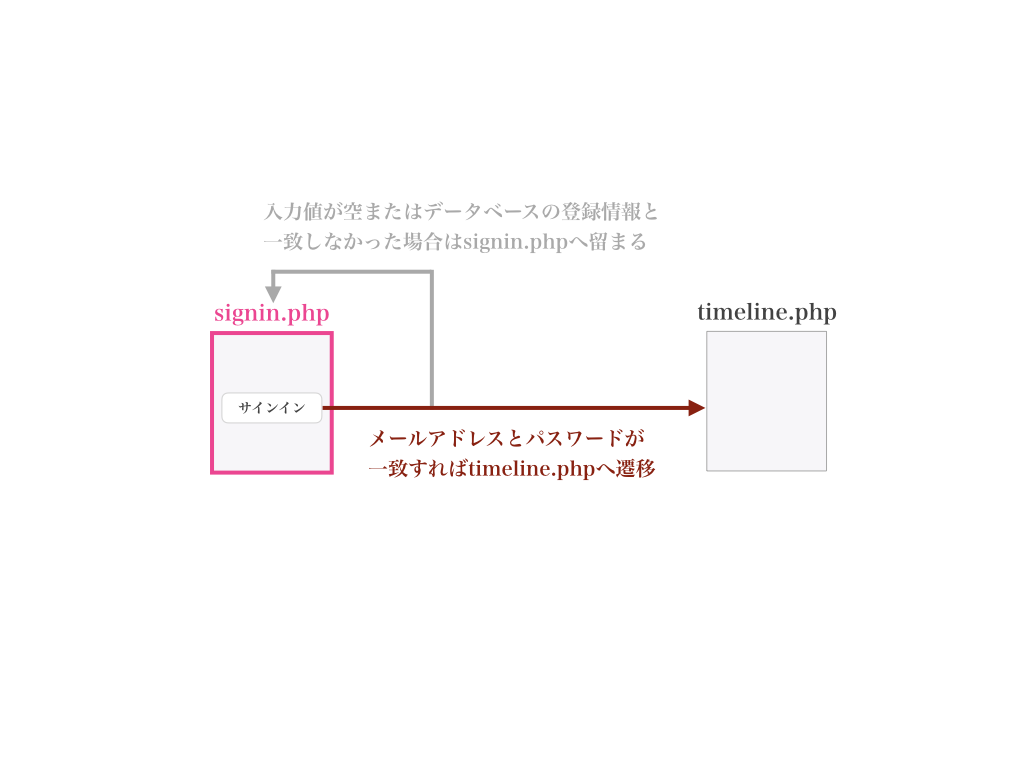
## **要件確認**

続いて、サインイン画面の要件を確認していきます。

### **要件リスト**

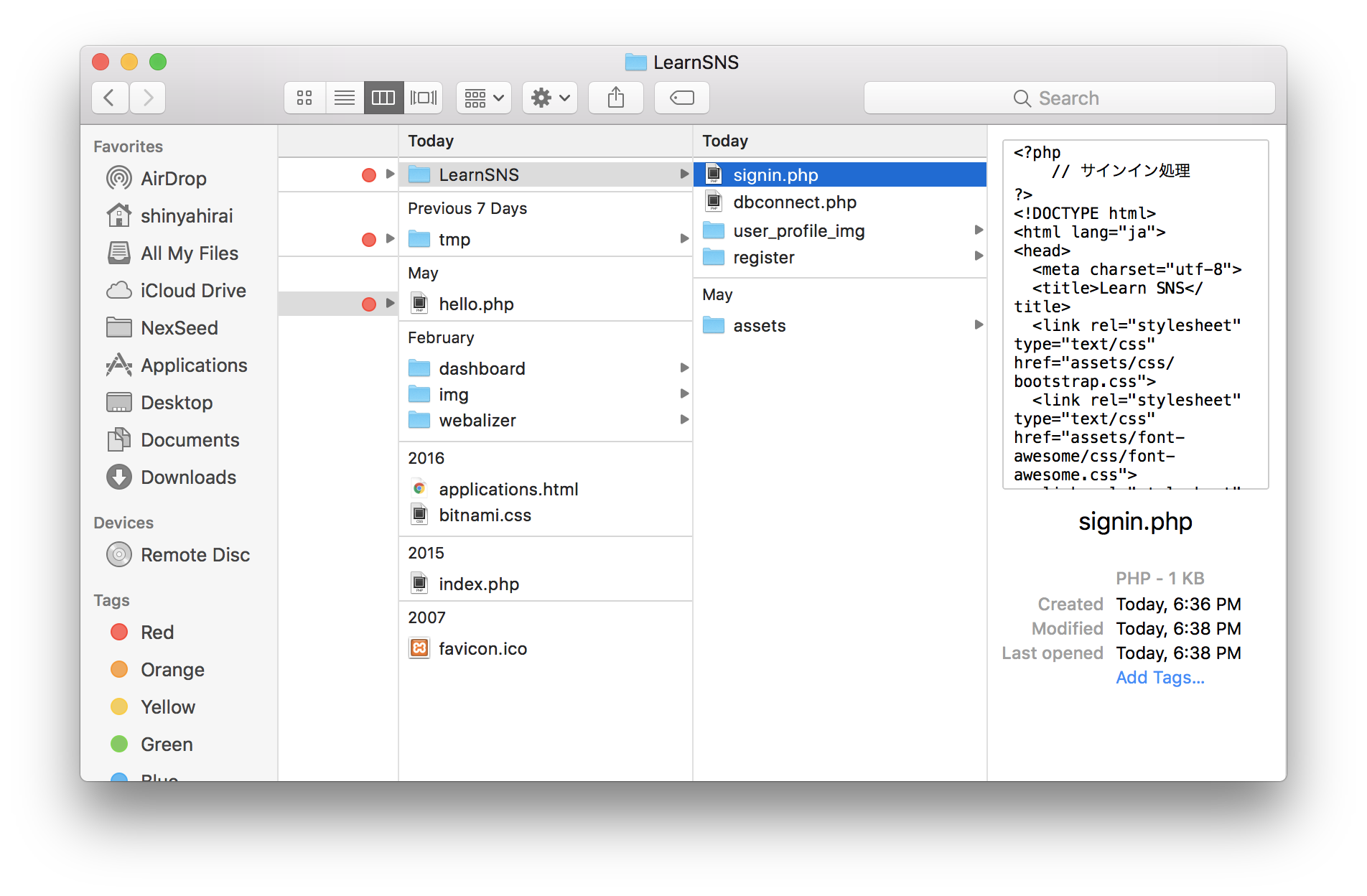
1. ユーザーは登録済み「メールアドレス」と「パスワード」の組み合わせを入力することができる
2. メールアドレスとパスワードが空だった場合のバリデーション（blank）がある
3. 入力されたメールアドレスと一致する登録データ1件をデータベースから取得する
4. 取得したデータのパスワードと入力されたパスワードをpassword\_verify()関数にかけて一致するかを検証する
5. 入力されたメールアドレスでデータが取得できないまたはパスワードが一致しない場合のバリデーション（failed）がある
6. パスワードが一致した場合はサインイン処理を実行する
   * サインインアカウントのidを$\_SESSION['LearnSNS']['id']に保存する
   * header()関数を使ってタイムライン画面へ遷移する

### **フローマップ**

****

## **サインイン画面の作成**

要件が確認できたので、signin.phpファイルの実装に入っていきます。



## **バリデーション実装**

まずは各種バリデーション機能から。

ここでは、要件で確認したとおり下記バリデーションを実装していきます。

* メールアドレスとパスワードが空だった場合のバリデーション（blank）がある
* 入力されたメールアドレスでデータが取得できないまたはパスワードが一致しない場合のバリデーション（failed）がある

### **空チェック**

ではまず**空チェック**から実装していきます。

実装の流れをまとめると下記。

1. エラーだった場合に何エラーかを保存するための$errors配列を定義
2. 送信されたデータと''（空文字）とを比較
3. 一致する場合は$errors配列にsigninキーをセットし、blankという値を保存
4. HTMLのメールアドレス入力フォーム下にもし$errors配列のsigninキーが存在し、blankという値が入っていた場合はエラーメッセージメールアドレスとパスワードを入力してくださいを出力

#### **1の処理**

「エラーだった場合に何エラーかを保存するための$errors配列を定義」

signin.phpに$errors配列の定義文を記述します。このとき、まだ何もエラーはでないため[]と空の配列として定義します。

##### **signin.php**

**<?php**  
 *// 初期化*  
 $errors **=** **[]**;  
**?>**  
<!DOCTYPE html>  
<html lang="ja">  
<head>  
・・・

#### **2, 3の処理**

「送信されたデータと''（空文字）とを比較」

「一致する場合は$errors配列にsigninキーをセットし、blankという値を保存」

**if文**を使用して送信されたメールアドレスとパスワードが空かどうかを比較します。

今回もformタグのmethod属性にはPOST送信指定がされています。action属性には自身(signin.php)を指定しましょう。サインインボタンを押した時に$\_POSTを生成した上でsignin.phpを再度読み直します。

##### **signin.php**

**<!DOCTYPE html>**  
<html lang**=**"ja">  
・・・  
 <form method**=**"POST" action**=**"signin.php">  
 <div class**=**"form-group">  
 <label for**=**"email">メールアドレス</label>  
 <input type**=**"email" name**=**"input\_email" class**=**"form-control" id**=**"email" placeholder**=**"example@gmail.com">  
 </div>  
 <div class**=**"form-group">  
 <label for**=**"password">パスワード</label>  
 <input type**=**"password" name**=**"input\_password" class**=**"form-control" id**=**"password" placeholder**=**"4 ~ 16文字のパスワード">  
 </div>  
 <input type**=**"submit" class**=**"btn btn-info" value**=**"サインイン">  
 </form>  
・・・  
</html>

メールアドレスはinputタグのname属性値がinput\_emailとなっているので、$\_POST['input\_email']、パスワードはinputタグのname属性値がinput\_passwordとなっているので、$\_POST['input\_password']で受け取ることができます。

これから記述する処理もユーザーがサインインボタンを押した時のみ処理したいプログラムです。 そのため、if(!empty($\_POST)){}を使ってフォーム送信時（サインインボタンを押したとき）のみ実行されるよう条件分岐します。

以上のことから、下記のようなプログラムを記述しメールアドレス / パスワード両フォームの空チェックプログラムを完成させます。

##### **signin.php**

**<?php**  
 *// 初期化*  
 $errors **=** **[]**;  
  
 **if** (**!empty**($\_POST)) {  
 *// ①*  
 $email **=** $\_POST['input\_email'];  
 $password **=** $\_POST['input\_password'];  
 **if** ($email **!=** '' **&&** $password **!=** '') {  
 *// データベースとの照合処理*  
 } **else** {  
 $errors['signin'] **=** 'blank';  
 }  
 }  
**?>**  
<!DOCTYPE html>  
・・・

コメントアウトで①としている部分は$\_POST['input\_email']と$\_POST['input\_password']を$email、$passwordという簡単な変数へ代入しています。

#### **4の処理**

「HTMLのメールアドレス入力フォーム下にもし$errors配列のsigninキーが存在し、blankという値が入っていた場合はエラーメッセージメールアドレスとパスワードを入力してくださいを出力」

このプログラムにより、ユーザーが各種入力フォームを空のままサインインボタンを押した際に$errors配列にはsigninキーとblankのペア情報が代入されます。この配列を利用し、画面上にエラーメッセージを表示するプログラムを記述します。

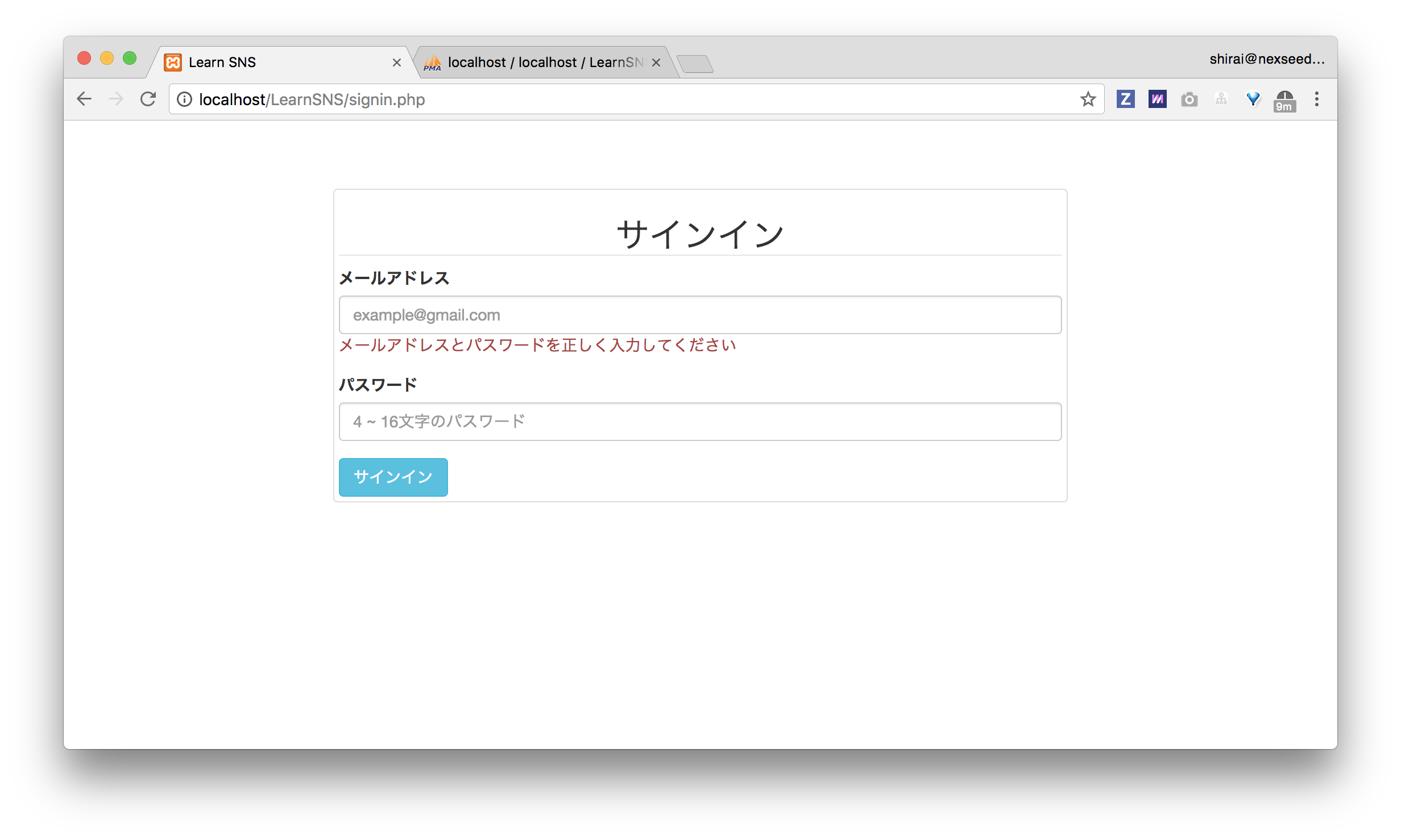
$errors['signin']にblankが代入されているとき = ユーザーが空のままサインインボタンを押したということになるので、$errors['signin']が存在し且つblankが代入されていればエラーメッセージがでるようにメールアドレスのinputタグ下にプログラムを追加します。

##### **signin.php**

<!DOCTYPE html>  
<html lang="ja">  
・・・  
 <form method="POST" action="signin.php">  
 <div class="form-group">  
 <label for="email">メールアドレス</label>  
 <input type="email" name="input\_email" class="form-control" id="email" placeholder="example@gmail.com">  
 **<?php if(isset($errors['signin']) && $errors['signin'] == 'blank'): ?>  
 <p class="text-danger">メールアドレスとパスワードを正しく入力してください</p>  
 <?php endif; ?>** </div>  
 <div class="form-group">  
 <label for="password">パスワード</label>  
 <input type="password" name="input\_password" class="form-control" id="password" placeholder="4 ~ 16文字のパスワード">  
 </div>  
 <input type="submit" class="btn btn-info" value="サインイン">  
 </form>  
・・・  
</html>

ここまでプログラムが完成したら、一度signin.phpをブラウザで開き入力フォームを空のままサインインボタンを押してみましょう。

入力フォームの下に赤色でエラーメッセージが出力されれば**成功**です。



## **空チェックバリデーション成功時の処理**

次は空チェックバリデーションを通過した（成功）場合の処理について進めていきます。

メールアドレスとパスワードの入力フォームが空でなければif文はtrueを返し{}内を処理するので、そこへ要件通り下記の処理をプログラムしていきます。 (もう一方のfailedバリデーションはこの処理内で実装します)

* 入力されたメールアドレスと一致する登録データ1件をデータベースから取得する
* 取得したデータのパスワードと入力されたパスワードをpassword\_verify()関数にかけて一致するかを検証する
* パスワードが一致した場合はサインイン処理を実行する
  + サインインアカウントのidを$\_SESSION['LearnSNS']['id']に保存する
  + header()関数を使ってタイムライン画面へ遷移する

#### **1. 入力されたメールアドレスと一致する登録データ1件をデータベースから取得する**

まずはデータベースへ接続し、メールアドレスを条件にデータを1件取得し$recordに代入します。

**<?php**  
 **require**('dbconnect.php');  
  
 *// 初期化*  
 $errors **=** **[]**;  
  
 **if** (**!empty**($\_POST)) {  
 $email **=** $\_POST['input\_email'];  
 $password **=** $\_POST['input\_password'];  
  
 **if** ($email **!=** '' **&&** $password **!=** '') {  
 *// ①*  
 $sql **=** 'SELECT \* FROM `users` WHERE `email`=?';  
 $data **=** **[**$email];  
 $stmt **=** $dbh**->**prepare($sql);  
 $stmt**->**execute($data);  
 $record **=** $stmt**->**fetch(PDO**::**FETCH\_ASSOC);  
  
 *// メールアドレスでの本人確認*  
 **if** ($record **==** **false**) {  
 $errors['signin'] **=** 'failed';  
 }  
 } **else** {  
 $errors['signin'] **=** 'blank';  
 }  
 }  
**?>**

コメントアウト①のSELECT文部分、WHERE句を使ってemailカラムの値と$emailが一致するという条件でデータを1件取得しています。

このSQL文の結果、データが一致すれば$recordにはアカウント情報が連想配列形式で代入されます。逆に入力されたメールアドレスが一致しなかった場合はfalseが代入されるので、その場合は入力値にミスがあったことをエラーメッセージで表示します(failedエラー)。

このプログラムにより、$errors配列にはsigninキーとfailedのペア情報が代入されます。

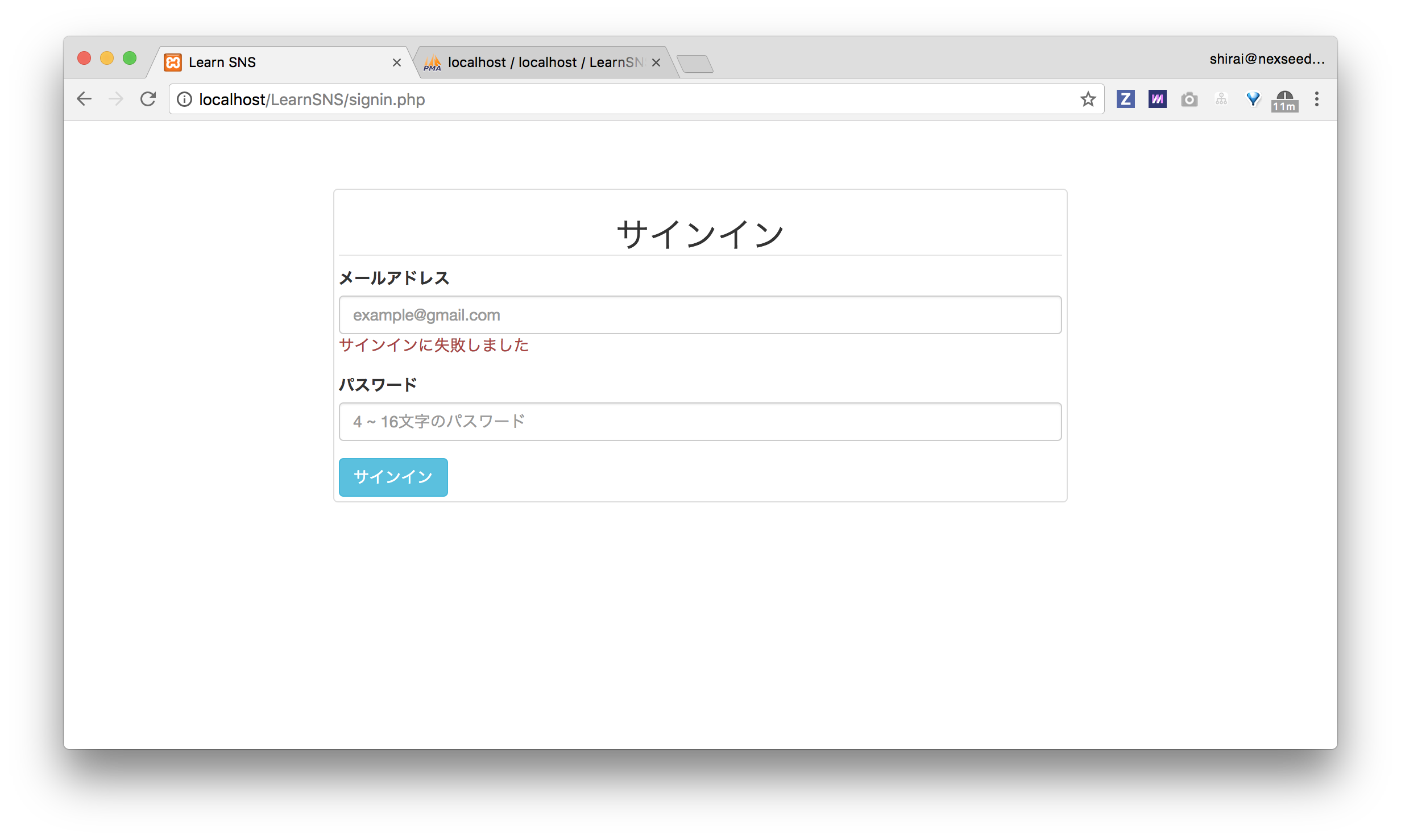
この配列を利用し、画面上にエラーメッセージを表示するプログラムを記述します。

##### **signin.php**

<!DOCTYPE html>  
<html lang="ja">  
・・・  
 <form method="POST" action="signin.php">  
 <div class="form-group">  
 <label for="email">メールアドレス</label>  
 <input type="email" name="input\_email" class="form-control" id="email" placeholder="example@gmail.com">  
 **<?php** **if**(isset($errors['signin']) **&&** $errors['signin'] **==** 'blank'): **?>**  
 <p class="text-danger">メールアドレスとパスワードを正しく入力してください</p>  
 **<?php** endif; **?>**  
 **<?php if(isset($errors['signin']) && $errors['signin'] == 'failed'): ?>  
 <p class="text-danger">サインインに失敗しました</p>  
 <?php endif; ?>**  
 </div>  
・・・  
</html>

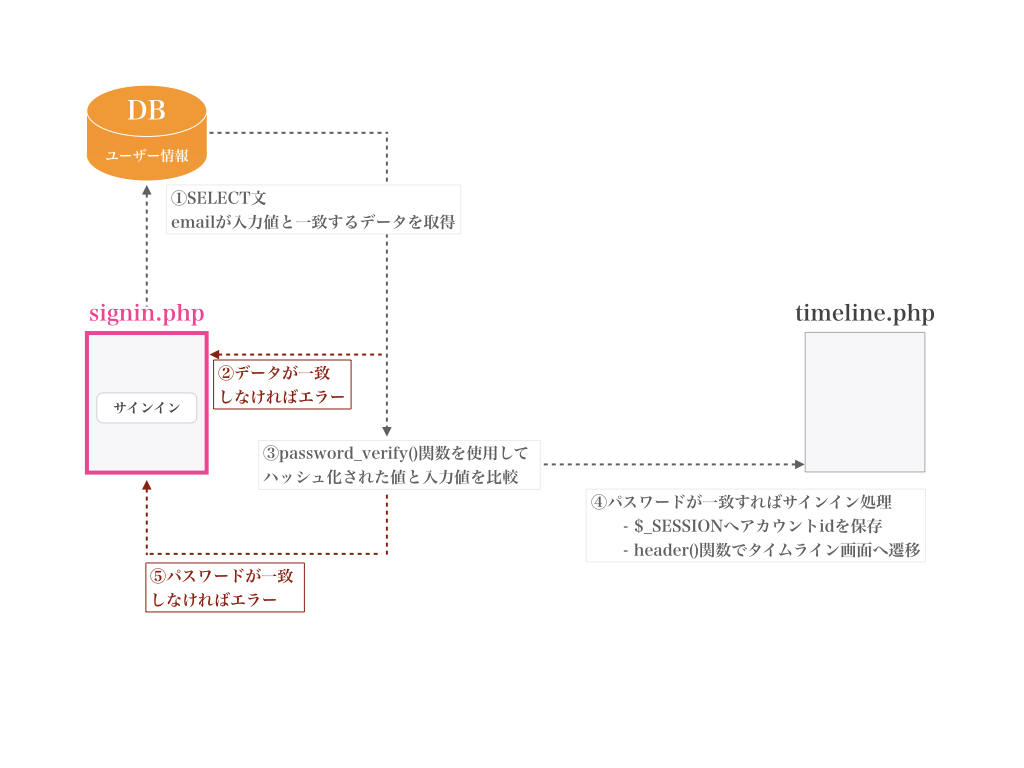
signin.phpをブラウザで開きメールアドレスに間違った値を入れてサインインボタンを押してみましょう。パスワードにも何かしらの値を入れておかなければ空チェックが先に実行されてしまうので注意してください。

入力フォームの下に赤色でエラーメッセージが出力されれば**成功**です。



#### **2. 取得したデータのパスワードと入力されたパスワードをpassword\_verify()関数にかけて一致するかを検証する**

ここから分岐処理が更に増え複雑なプログラムになってくるので、先に下記図で流れと分岐のパターンをまとめておきます。



今までのプログラムで①②は実装済みなので、ここからは③の処理を実装していきます。

SELECT文で取得したデータが代入されている$record連想配列からパスワードカラムのデータを取得し、ハッシュ化されたパスワードと、入力されたパスワードをpassword\_verify()を使って比較します。

##### **signin.php**

...

if (password\_verify($password,$record['password'])){  
 //認証成功  
}else{

//認証失敗

$errors['signin'] = 'failed';

}  
...

認証に失敗した場合は、事前に設定していた、入力されたemailが登録されていない場合と同じ$errors配列にはsigninキーとfailedのペア情報を代入して認証失敗のエラーを表示しておきます。

#### **3. パスワードが一致した場合はサインイン処理を実行する**

認証が晴れて成功した場合、以下の処理を行います。

* サインインアカウントのidを$\_SESSION['LearnSNS']['id']に保存する
* header()関数を使ってタイムライン画面へ遷移する

認証が成功した際のアカウントのidは重要な情報です。ログイン後あらゆる画面でログインユーザーを特定するために使用されます。他の情報は別のタイミングでユーザー自身により編集される可能性があるため、ログイン処理が存在するサービスでは通常idだけを常にSESSION変数に保持するという対策を取ります。

##### **signin.php**

session\_start();

...

if (password\_verify($password,$record['password'])){

//認証成功

//※追加部分

//SESSION変数にIDを保存

$\_SESSION['LearnSNS']['id'] = $record['id'];

//timeline.phpに移動

header("Location: timeline.php");

exit();

}else{

$errors['signin'] = 'failed';

}

…

LearnSNS/timeline.php

<?php

session\_start();

var\_dump($\_SESSION);

echo ‘とんだよ’;

?>