1. **Защита от XSS**

Проверяем все данный поступившие извне на соответствие формату, в то числе с помощью preg_match.

```php
if (empty($biography)  ) {
  setcookie('biography_error', '1', time() + 24 * 60 * 60);
  $errors = TRUE;
}
else if(!preg_match('/^[a-zA-Za-яА-яёЁ0-9.,;!? \-]+$/u', $biography) ||strlen($biography) > 128){
  if(!preg_match('/^[a-zA-Za-яА-яёЁ0-9.,;!? \-]+$/u', $biography)){
  setcookie('biography_error', '2', time() + 24 * 60 * 60);
  $errors = TRUE;}
  else if(strlen($biography) > 128){
    setcookie('biography_error', '3', time() + 24 * 60 * 60);
    $errors = TRUE;}
```

```php
if (empty($email) ) {
  setcookie('email_error', '1', time() + 24 * 60 * 60);
  $errors = TRUE;
}
else if(!filter_var($email, FILTER_VALIDATE_EMAIL)){
  setcookie('email_error', '2', time() + 24 * 60 * 60);
  $errors = TRUE;
}
```

htmlspecialchars — Преобразовывает специальные символы в HTML-сущности, чтобы браузер гарантированно воспринимал строку как строку и не пытался выполнить ее как код

```php
$name = htmlspecialchars($_POST['name'], ENT_QUOTES, 'UTF-8');
$phone = htmlspecialchars($_POST['phone'], ENT_QUOTES, 'UTF-8');
$email = htmlspecialchars($_POST['email'], ENT_QUOTES, 'UTF-8');
$date = $_POST['date'];
```

2. **Защита от Information Disclosure**

Пароли не хранятся в открытом коде, есть только на сервере

```php
include('../data.php');
$db = new PDO('mysql:host=localhost;dbname=u67325', $db_user, $db_pass,
  [PDO::ATTR_PERSISTENT => true, PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
// Проверяем меняются ли ранее сохраненные данные или отправляются новые.
```

3. **Защита от SQL Injection**

Используем подготовленные запросы (Prepared Statements, PDO)

```php
    $sth = $db->prepare('SELECT id FROM login_and_password WHERE login = :login');
    $sth->execute(['login' => $userLogin]);

    while ($row = $sth->fetch()) {
      $formId = $row['id'];
    }

$stmt = $db->prepare("UPDATE users SET name = :name, phone = :phone, email = :email, date=:date,  gender = :gender, biography = :biogra
$stmt -> execute(['name'=>$name,'phone'=>$phone, 'email'=>$email,'date'=>$date,'gender'=>$gender,'biography'=>$biography, 'checkboxCont

$stmt = $db->prepare("DELETE FROM users_and_languages WHERE id_user = :formId");
$stmt -> execute(['formId'=>$formId]);
$stmt = $db->prepare("INSERT INTO users_and_languages (id_user, id_lang) VALUES (:id_user, :id_lang)");
    foreach ($_POST['favourite_lan'] as $id_lang) {
        $stmt->bindParam(':id_user', $id_user);
        $stmt->bindParam(':id_lang', $id_lang);
        $id_user = $formId;
        $stmt->execute();
    }
```

4. Защита от CSRF

Anti-CSRF токен. Сервер генерирует случайный уникальный токен для каждого сеанса пользователя и проверяет его для каждого запроса. Если токены не совпадают, запрос отклоняется.

```php
  $token = uniqid();
  setcookie('token', $token, time() + 24 * 60 * 60);

  include('index1.php');

}
else {
    if($_POST('token')!=$_COOKIE('token'){
      exit();
    })
```

```php
<input type="hidden" name="token" value = "<?=$token?>">
```

5. Защита от Include

Все include запросы находятся со стороны сервера, у пользователя нет доступа к ним.

Реализована аутентификация пользователей, только конкретный пользователь может менять как-либо свои данные.

6. Защита от Upload

Upload уязвимости нет, тк нет возможности загрузить файлы.