VerifyLens Landing Page - Registration Integration

Overview

This document describes the integration of user registration into the VerifyLens landing page credits purchase flow.

What Changed

New Components

1. Registration Modal (`/components/registration-modal.tsx`)

A comprehensive registration form modal with:
- **Fields**: First Name, Last Name, Email, Phone Number, Company Name, Password, Confirm Password
- **Validation**: Real-time validation with error messages
- **Password Strength**: Visual indicator showing password strength (5 levels)
- **Show/Hide Password**: Toggle visibility for password fields
- **Loading States**: Disabled inputs and loading spinner during submission
- **Error Handling**: Displays API errors with toast notifications

Props:
- `isOpen`: Boolean to control modal visibility
- `onClose`: Callback when modal is closed
- `onSuccess`: Callback with user data (userId, customerId, email) on successful registration
- `packageInfo`: Optional package details to display in modal

Modified Files

1. Credits Page (`/app/credits/page.tsx`)

Added:
- Import for `RegistrationModal` component
- State management for registration modal
- State to store registration data (userId, customerId, email)
- `handleRegistrationSuccess` function to handle post-registration checkout

Updated:
- `handlePurchase` function now opens registration modal instead of going directly to checkout
- After registration, automatically proceeds to Stripe checkout with user data
- Passes package information to registration modal

2. Checkout API (`/app/api/checkout/route.ts`)

Updated:
- Now requires `userId` parameter (in addition to email and customerId)
- Updated validation to enforce all three parameters
- Passes `user_id` and `customer_id` to Stripe metadata
- Stripe webhook will receive these IDs to allocate credits to the correct user/customer

New Environment Variables

Added to `.env.example`:
- `NEXT_PUBLIC_APP_URL`: URL of main app (where registration API is hosted)
- `NEXT_PUBLIC_SITE_URL`: URL of landing page (this site)

Required for Vercel:

```
NEXT_PUBLIC_APP_URL=https://www.verifylens.com
```

User Flow

1. **User Visits Credits Page**
   - Views credit packages
   - Clicks "Purchase Package" button

2. **Registration Modal Opens**
   - User fills in registration form
   - Form validates inputs in real-time
   - Password strength indicator shows security level
   - User submits form

3. **Registration API Call**
   - POST request to `${NEXT_PUBLIC_APP_URL}/api/auth/register`
   - Creates customer and user in main app database
   - Sends verification email to user
   - Returns userId and customerId

4. **Success Handling**
   - Modal closes automatically
   - Toast notification shows success message
   - Automatically proceeds to Stripe checkout

5. **Stripe Checkout**
   - POST request to `/api/checkout` with userId, customerId, email
   - Creates Stripe checkout session with metadata
   - Redirects user to Stripe payment page

6. **Payment Completion**
   - User completes payment on Stripe
   - Stripe webhook notifies main app
   - Main app allocates credits to user's account
   - Welcome email sent with login link

7. **Email Verification**
   - User receives verification email
   - Clicks verification link
   - Email marked as verified in database

8. **User Logs In**
   - User clicks login link from welcome email

- Signs in to main app with credentials
- Can now use credits for searches

API Integration

Registration API

Endpoint: `POST ${NEXT_PUBLIC_APP_URL}/api/auth/register`

Request:

```
{
  firstName: string;      // Required, 1-100 chars
  lastName: string;       // Required, 1-100 chars
  email: string;          // Required, valid email format
  phoneNumber?: string;   // Optional
  companyName: string;    // Required, 1-255 chars
  password: string;       // Required, min 8 chars with complexity
}
```

Response:

```
{
  success: boolean;
  userId: number;
  customerId: number;
  email: string;
  message: string;
}
```

Error Response:

```
{
  success: false;
  error: string;
}
```

Checkout API

Endpoint: `POST /api/checkout`

Request:

```
{
  packageName: string;
  credits: number;
  price: number;        // In cents
  email: string;        // From registration
  customerId: number;   // From registration
  userId: number;       // From registration
}
```

Response:

```
{
  sessionId: string;
  url: string;         // Stripe checkout URL
}
```

Installation & Setup

1. Install Dependencies

```
cd /home/ubuntu/github_repos/roblox-lander
npm install
```

2. Update Environment Variables

Local Development (`.env`):

```
NEXT_PUBLIC_APP_URL=http://localhost:3000
STRIPE_SECRET_KEY=sk_test_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_...
```

Production (Vercel):

```
NEXT_PUBLIC_APP_URL=https://www.verifylens.com
STRIPE_SECRET_KEY=sk_live_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_...
```

3. Deploy to Vercel

```
git add .
git commit -m "feat: Add registration modal and checkout integration"
git push origin main
```

Vercel will automatically deploy the changes.

Testing

Local Testing

1. Start the development server:

```
npm run dev
```

1. Navigate to `/credits`
2. Click “Purchase Package”
3. Fill in registration form with test data:
   - First Name: Test
   - Last Name: User
   - Email: test@example.com

   - Company: Test Law Firm
   - Password: Test1234!

4. Submit form and verify:
   - Success toast appears
   - Redirects to Stripe checkout
   - Stripe shows correct package details

Production Testing

1. Use Stripe test mode keys
2. Test card: `4242 4242 4242 4242`
3. Use any future expiry date and any CVC
4. Complete checkout
5. Verify:
   - Webhook processes payment
   - Credits added to account
   - Welcome email received
   - Verification email received
   - Can log in with credentials

Error Handling

The registration modal handles various error scenarios:

Validation Errors

- Empty required fields
- Invalid email format
- Weak password
- Password mismatch

API Errors

- Duplicate email (409)
- Duplicate company name (409)
- Server errors (500)
- Network errors

All errors are displayed to the user via:
- Inline field errors (red text with alert icon)
- Toast notifications (for API errors)

Password Requirements

- Minimum 8 characters
- At least one uppercase letter (A-Z)
- At least one lowercase letter (a-z)
- At least one number (0-9)
- At least one special character (!@#$%^&*()_+-=[]{}|;:,.<>?)

Password strength levels:
1. **Very Weak** (1/5): Missing multiple requirements

2. **Weak** (2/5): Missing some requirements
3. **Fair** (3/5): Meets basic requirements
4. **Good** (4/5): Strong password
5. **Strong** (5/5): Very strong password

Troubleshooting

Modal Doesn’t Open

- Check that registration modal is imported correctly
- Verify `showRegistrationModal` state is toggled on click
- Check browser console for errors

Registration Fails

- Verify `NEXT_PUBLIC_APP_URL` is set correctly
- Check main app is running and accessible
- Verify registration API endpoint exists
- Check network tab for request/response details

Checkout Fails

- Verify userId and customerId are passed correctly
- Check Stripe keys are valid
- Verify checkout API receives all required params
- Check Stripe dashboard for errors

Credits Not Added

- Check webhook is configured in Stripe dashboard
- Verify webhook secret matches
- Check main app logs for webhook processing
- Verify metadata is included in payment_intent_data

Code Examples

Using the Registration Modal

```
import RegistrationModal from '@/components/registration-modal'

function MyComponent() {
  const [showModal, setShowModal] = useState(false)

  const handleSuccess = (data) => {
    console.log('User registered:', data)
    // Proceed with checkout using data.userId, data.customerId, data.email
  }

  return (
    <>
      <button onClick={() => setShowModal(true)}>
        Register
      </button>

      <RegistrationModal
        isOpen={showModal}
        onClose={() => setShowModal(false)}
        onSuccess={handleSuccess}
        packageInfo={{
          name: "Professional",
          credits: 50,
          price: 5000 // in cents
        }}
      />
    </>
  )
}
```

Calling Registration API

```
const response = await fetch(`${process.env.NEXT_PUBLIC_APP_URL}/api/auth/register`, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    firstName: 'John',
    lastName: 'Doe',
    email: 'john@example.com',
    phoneNumber: '+1 555-0123',
    companyName: 'ABC Law Firm',
    password: 'SecurePass123!',
  }),
})

const data = await response.json()

if (!response.ok) {
  throw new Error(data.error || 'Registration failed')
}

// data contains: { success, userId, customerId, email, message }
```

Security Notes

1. **Password Validation**: Done on both client and server
2. **HTTPS Required**: All API calls use HTTPS in production
3. **Input Sanitization**: All inputs are trimmed and validated
4. **Error Messages**: Generic errors shown to prevent information disclosure
5. **Rate Limiting**: Consider implementing on registration endpoint
6. **CAPTCHA**: Consider adding to prevent automated registrations

Support

For issues or questions:
- Main Documentation: See `REGISTRATION_CHECKOUT_IMPLEMENTATION.md` in main app
- Email: support@verifylens.com

Last Updated: October 29, 2025
Version: 1.0