VerifyLens Registration & Checkout Implementation

Overview

This document describes the complete registration and checkout flow implemented for VerifyLens, allowing users to create accounts from the landing page and proceed to purchase credits with Stripe integration.

Architecture

Flow Diagram

```
Landing Page (roblox-lander)
  ↓ User clicks "Purchase Package"
  ↓
Registration Modal (New User Registration)
  ↓ Submit registration form
  ↓
Main App API (/api/auth/register)
  ↓ Create customer & user in database
  ↓ Send verification email
  ↓
Return userId & customerId
  ↓
Lander Checkout API (/api/checkout)
  ↓ Create Stripe checkout session with metadata
  ↓
Stripe Checkout Page
  ↓ User completes payment
  ↓
Stripe Webhook (/api/credits/webhook)
  ↓ Verify payment
  ↓ Add credits to customer account
  ↓ Send welcome email with login link
  ↓
User logs in to Main App
```

Implementation Details

1. Database Schema Changes

File: `/database/migrations/002_add_email_verification.sql`

Added fields to `users` table:
- `email_verified` (BOOLEAN, default false)
- `email_verification_token` (VARCHAR(255), nullable)
- `email_verification_expires` (TIMESTAMP, nullable)
- `phone_number` (VARCHAR(50), nullable)

To Apply Migration:

```
psql $DATABASE_URL -f database/migrations/002_add_email_verification.sql
```

2. Registration API Endpoint

File: `/src/app/api/auth/register/route.ts`

Endpoint: `POST /api/auth/register`

Request Body:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@lawfirm.com",
  "phoneNumber": "+1 (555) 123-4567",
  "companyName": "Smith & Associates",
  "password": "SecurePass123!"
}
```

Response:

```
{
  "success": true,
  "userId": 123,
  "customerId": 456,
  "email": "john.doe@lawfirm.com",
  "message": "Account created successfully. Please check your email to verify your ac-
count."
}
```

Features:
- Email format validation
- Password complexity validation (min 8 chars, uppercase, lowercase, number, special char)
- Duplicate email/company check
- Creates both customer and user records in a transaction
- Generates email verification token (24-hour expiry)
- Sends verification email via Resend

3. Email Verification Endpoint

File: `/src/app/api/auth/verify-email/route.ts`

Endpoint: `GET /api/auth/verify-email?token={token}`

Features:
- Validates verification token
- Checks token expiration (24 hours)
- Marks email as verified
- Redirects to sign-in page with appropriate message

Redirect URLs:
- Success: `/auth/signin?message=EmailVerified`
- Already Verified: `/auth/signin?message=AlreadyVerified`

- Invalid Token: `/auth/signin?error=InvalidToken`
- Expired Token: `/auth/signin?error=TokenExpired`

4. Email Templates

File: `/src/app/lib/email/index.ts`

Added Email Types:

Verification Email

- Sent immediately after registration
- Contains clickable verification link
- 24-hour expiration notice
- Security notice about unwanted registrations

Welcome Email (Updated)

- Sent after successful payment
- Shows username for login
- Shows password (if temp password) or login instructions
- Displays credit balance
- Includes login link

5. Webhook Updates

File: `/src/app/api/credits/webhook/route.ts`

Changes:
- Added `email_verified` field to customer query
- Updated first purchase detection logic
- Modified email sending logic:
- If `userId` and `customerId` are present (new flow): Send welcome email without temp password
- If only `customerId` present (legacy): Generate temp password and send welcome email
- Otherwise: Send purchase confirmation email

Metadata Required:
- `customer_id`: Customer ID from registration
- `user_id`: User ID from registration
- `credits`: Number of credits purchased
- `package_id`: Package identifier (optional)

6. Landing Page Components

Registration Modal

File: `/components/registration-modal.tsx`

Features:
- Clean, modern UI with validation
- Real-time password strength indicator
- Show/hide password toggles
- Form validation with error messages
- Loading states during registration
- Package information display
- Security notices

Credits Page Updates

File: `/app/credits/page.tsx`

Changes:
- Added registration modal integration
- Updated purchase flow:
1. Show registration modal
2. Create account
3. Proceed to checkout with user data
- Pass userId and customerId to checkout API

Checkout API Updates

File: `/app/api/checkout/route.ts`

Changes:
- Now requires `email`, `customerId`, and `userId`
- Passes metadata to Stripe session
- Includes metadata in `payment_intent_data` for webhook

Environment Variables

Main App (roblox-tool)

Required in Vercel/Environment:

```
# Database
DATABASE_URL=postgresql://...

# NextAuth
NEXTAUTH_SECRET=your-secret-key
NEXTAUTH_URL=https://www.verifylens.com
APP_URL=https://www.verifylens.com

# Stripe
STRIPE_SECRET_KEY=sk_live_...
STRIPE_WEBHOOK_SECRET=whsec_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_...

# Resend Email
RESEND_API_KEY=re_...

# Redis (if using)
REDIS_URL=redis://...
```

Landing Page (roblox-lander)

Required in Vercel/Environment:

```
# Stripe
STRIPE_SECRET_KEY=sk_live_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_...

# Main App URL
NEXT_PUBLIC_APP_URL=https://www.verifylens.com

# Lander URL
NEXT_PUBLIC_SITE_URL=https://site.verifylens.com
```

Deployment Steps

1. Apply Database Migration

```
# Connect to your Supabase/PostgreSQL database
psql $DATABASE_URL -f database/migrations/002_add_email_verification.sql
```

2. Update Environment Variables

Main App (Vercel):

1. Go to Vercel dashboard for roblox-tool
2. Settings → Environment Variables
3. Add `NEXT_PUBLIC_APP_URL` if not present
4. Verify `RESEND_API_KEY` is set
5. Verify `STRIPE_WEBHOOK_SECRET` is set

Landing Page (Vercel):

1. Go to Vercel dashboard for roblox-lander
2. Settings → Environment Variables
3. Add `NEXT_PUBLIC_APP_URL` = `https://www.verifylens.com`
4. Verify Stripe keys are set

3. Deploy Both Applications

```
# Main App
cd /home/ubuntu/github_repos/roblox-tool
git add .
git commit -m "feat: Implement registration and checkout flow"
git push origin main

# Landing Page
cd /home/ubuntu/github_repos/roblox-lander
git add .
git commit -m "feat: Add registration modal and checkout integration"
git push origin main
```

Vercel will automatically deploy both applications.

4. Test Stripe Webhook

Use Stripe CLI to test webhook locally:

```
stripe listen --forward-to https://www.verifylens.com/api/credits/webhook
stripe trigger checkout.session.completed
```

Testing Checklist

Registration Flow

- [ ] Open landing page credits section
- [ ] Click “Purchase Package”
- [ ] Fill in registration form with valid data
- [ ] Submit form
- [ ] Verify success message appears
- [ ] Check email for verification link
- [ ] Click verification link
- [ ] Verify redirect to sign-in page with success message

Checkout Flow

- [ ] After registration, automatically redirected to Stripe checkout
- [ ] Verify package details are correct
- [ ] Complete test payment (use Stripe test card: 4242 4242 4242 4242)
- [ ] Verify redirect to success page
- [ ] Check email for welcome email
- [ ] Verify welcome email contains username and login link
- [ ] Click login link
- [ ] Sign in with username and password set during registration
- [ ] Verify credits are added to account

Error Cases

- [ ] Test duplicate email registration
- [ ] Test duplicate company name registration
- [ ] Test invalid email format
- [ ] Test weak password
- [ ] Test password mismatch
- [ ] Test expired verification token
- [ ] Test invalid verification token
- [ ] Test checkout without registration
- [ ] Test webhook with missing metadata

Security Considerations

1. **Password Storage**: Passwords are hashed with bcrypt (10 rounds)
2. **Email Verification**: Required before account can be used
3. **Token Expiration**: Verification tokens expire after 24 hours
4. **HTTPS Only**: All communications over HTTPS
5. **Stripe Integration**: PCI-compliant payment processing
6. **Input Validation**: All inputs validated on backend

7. **SQL Injection Protection**: Parameterized queries used throughout
8. **Rate Limiting**: Consider implementing rate limiting on registration endpoint

Monitoring & Logs

Key Log Points

1. **Registration**: `[Registration]` prefix
2. **Email Verification**: `[Email Verification]` prefix
3. **Webhook**: `[Webhook]` prefix
4. **Email Sending**: `[Email]` prefix

What to Monitor

- Registration success/failure rates
- Email verification rates
- Checkout completion rates
- Webhook processing errors
- Email delivery failures

Troubleshooting

Registration Fails

- Check database connection
- Verify required fields are provided
- Check for duplicate email/company name
- Review server logs for errors

Verification Email Not Received

- Check Resend dashboard for delivery status
- Verify RESEND_API_KEY is correct
- Check spam folder
- Verify email address is valid

Credits Not Added After Payment

- Check Stripe webhook is configured correctly
- Verify STRIPE_WEBHOOK_SECRET matches
- Check webhook logs in Stripe dashboard
- Verify metadata is being passed correctly
- Check database for customer_stats table

User Can’t Login

- Verify email was verified
- Check username is correct (generated from email prefix)
- Verify password was set correctly during registration
- Check user.is_active is true
- Check customer.is_active is true

Future Enhancements

1. **Password Reset Flow**: Add forgot password functionality
2. **Email Resend**: Allow users to resend verification email
3. **Social Login**: Add Google/Microsoft OAuth
4. **Two-Factor Authentication**: Add 2FA option for enhanced security
5. **Rate Limiting**: Implement rate limiting on registration
6. **CAPTCHA**: Add CAPTCHA to prevent bot registrations
7. **Email Preferences**: Allow users to opt-in/out of marketing emails
8. **Account Management**: Add profile editing functionality

Support

For issues or questions:
- Email: support@verifylens.com
- Documentation: This file
- Stripe Docs: https://stripe.com/docs
- Resend Docs: https://resend.com/docs

Last Updated: October 29, 2025
Author: DeepAgent (Abacus.AI)
Version: 1.0