perplexity

▯ Next Steps: Complete CURP-to-PDF Automation Guide

▯ Current Status

✅ **Completed:**

- Outlook account creation (up to captcha/long press)
- IMSS app automation (complete workflow)

▯ **Next Phase:** Email monitoring, PDF extraction, and workflow orchestration

▯ Sequential Implementation Steps

Phase 1: Complete Outlook Account Creation ▯ *2-3 days*

Step 1.1: Fix Account Creation Completion

```
# Add to your existing Outlook automation
def complete_account_verification(self):
    """Complete account creation after captcha"""
    # Wait for account creation confirmation
    # Verify account is fully active
    # Test login capability
    return account_details

def test_account_login(self, email, password):
    """Verify account can be logged into"""
    # Quick login test to confirm account works
    # Return success/failure status
```

Step 1.2: Database Integration

```
# Create database schema
def setup_database():
    """Create the 4 tables we designed"""
    # outlook_accounts table
    # imss_processing table
    # email_pdf_processing table
    # master_process_status table
```

```
def store_outlook_account(self, account_data):
    """Store created account in database"""
    # INSERT into outlook_accounts
    # SET status = 'completed'
    # Generate process_id for tracking
```

Phase 2: Email Monitoring System □ *4-5 days*

Step 2.1: Microsoft Graph API Setup

```
# Install required packages
pip install msal requests asyncio aiohttp
```

```
# Microsoft Graph API integration
class OutlookEmailMonitor:
    def __init__(self, email, password):
        self.email = email
        self.password = password
        self.access_token = None

    def authenticate(self):
        """Get OAuth token for Graph API"""
        # Microsoft Graph authentication
        # Store access token

    def check_inbox(self):
        """Check for new emails"""
        # GET /me/messages
        # Filter for IMSS emails
        # Return email list
```

Step 2.2: Email Polling Logic

```
def start_email_monitoring(self, process_id, email, curp_id):
    """Start monitoring email for IMSS response"""
    # INSERT into email_pdf_processing
    # SET status = 'monitoring'
    # Start polling loop

def poll_email_inbox(self, email_config):
    """Continuous email polling"""
    while True:
        emails = self.check_inbox()
        for email in emails:
            if self.is_imss_email(email):
                pdf_link = self.extract_pdf_link(email)
                if pdf_link:
                    return pdf_link
        time.sleep(60)  # Wait 1 minute between checks
```

Step 2.3: Email Parsing

```
def is_imss_email(self, email):
    """Check if email is from IMSS"""
    imss_senders = ['imss.gob.mx', 'noreply@imss', 'constancia@imss']
    return any(sender in email['from'] for sender in imss_senders)

def extract_pdf_link(self, email):
    """Extract PDF download link from email"""
    # Parse email body (HTML/text)
    # Look for download links
    # Validate PDF URLs
    # Return download URL
```

Phase 3: PDF Download & Processing *2-3 days*

Step 3.1: PDF Download

```
class PDFProcessor:
    def download_pdf(self, pdf_url, curp_id):
        """Download PDF from IMSS link"""
        response = requests.get(pdf_url)
        filename = f"{curp_id}.pdf"

        with open(f"pdfs/{filename}", 'wb') as f:
            f.write(response.content)

        return self.verify_pdf(filename)

    def verify_pdf(self, filename):
        """Verify PDF is valid and complete"""
        # Check file size > 0
        # Verify PDF format
        # Calculate SHA256 hash
        # Return metadata
```

Step 3.2: File Storage

```
def store_pdf_metadata(self, process_id, pdf_info):
    """Update database with PDF information"""
    # UPDATE email_pdf_processing
    # SET pdf_filename = '{curp_id}.pdf'
    # SET pdf_file_path, file_size, file_hash
    # SET status = 'completed'
```

Phase 4: Workflow Orchestrator □ *3-4 days*

Step 4.1: Database-Driven Orchestrator

```
class CURPWorkflowOrchestrator:
    def __init__(self):
        self.db = DatabaseConnection()
        self.outlook_bot = OutlookAutomator()
        self.imss_bot = IMSSAutomator()
        self.email_monitor = OutlookEmailMonitor()
        self.pdf_processor = PDFProcessor()

    def run_orchestrator_loop(self):
        """Main workflow loop"""
        while True:
            # Step 1: Process pending Outlook creations
            self.process_outlook_queue()

            # Step 2: Process completed Outlook → IMSS
            self.process_imss_queue()

            # Step 3: Process completed IMSS → Email monitoring
            self.process_email_queue()

            # Step 4: Check active email monitoring
            self.check_email_monitoring()

            time.sleep(30)  # 30 second cycle
```

Step 4.2: Queue Processing Methods

```
def process_outlook_queue(self):
    """Process pending Outlook account creations"""
    pending = self.db.get_pending_outlook_creations()

    for process in pending:
        try:
            result = self.outlook_bot.create_account(
                process['first_name'],
                process['last_name'],
                process['date_of_birth']
            )
            if result['success']:
                self.db.mark_outlook_completed(
                    process['process_id'],
                    result['email'],
                    result['password']
                )
        except Exception as e:
            self.db.mark_outlook_failed(process['process_id'], str(e))

def process_imss_queue(self):
    """Process Outlook accounts ready for IMSS"""
    ready_for_imss = self.db.get_ready_for_imss()

    for process in ready_for_imss:
        try:
            result = self.imss_bot.run_automation(
                process['curp_id'],
                process['email']
            )
            if result:
                self.db.mark_imss_completed(process['process_id'])
        except Exception as e:
            self.db.mark_imss_failed(process['process_id'], str(e))
```

Phase 5: Integration & Testing *2-3 days*

Step 5.1: End-to-End Testing

```
def test_complete_workflow():
    """Test complete CURP-to-PDF workflow"""

    # Test data
    test_curp = "TEST123456HEFGHI01"
    test_name = ("TestUser", "Demo")
    test_dob = "1990-01-01"

    # Start process
    orchestrator = CURPWorkflowOrchestrator()
    process_id = orchestrator.start_new_process(
        test_curp, test_name[0], test_name[1], test_dob
    )

    # Monitor progress
    while True:
        status = orchestrator.get_process_status(process_id)
        print(f"Status: {status['overall_status']} - {status['progress_percentage']}%")

        if status['overall_status'] in ['completed', 'failed']:
            break
        time.sleep(30)
```

Step 5.2: Error Handling & Fallbacks

```
def implement_fallback_mechanisms():
    """Add robust error handling"""

    # Retry logic for each stage
    # Fallback mechanisms
    # Error logging and reporting
    # Process recovery after failures
```

󰀀 Recommended Implementation Order

Week 1: Email Foundation

- ✅ Fix Outlook account completion
- ✅ Set up Microsoft Graph API
- ✅ Basic email polling

Week 2: PDF Processing

- ✅ Email parsing and link extraction
- ✅ PDF download and verification
- ✅ File storage system

Week 3: Database Integration

- ✅ Implement database schema
- ✅ Connect all components to database
- ✅ Basic orchestrator logic

Week 4: Orchestration & Testing

- ✅ Complete workflow orchestrator
- ✅ End-to-end testing
- ✅ Error handling and fallbacks

󰀀 Key Dependencies to Install

```
# Email & API
pip install msal requests aiohttp

# Database
pip install sqlalchemy alembic psycopg2-binary

# PDF Processing
pip install PyPDF2 requests

# Async Processing
pip install asyncio celery redis

# Monitoring
pip install prometheus-client

# Testing
pip install pytest pytest-asyncio factory-boy
```

Success Milestones

- ✅ **Milestone 1**: Complete Outlook account creation with database storage
- ✅ **Milestone 2**: Email monitoring successfully detects IMSS emails
- ✅ **Milestone 3**: PDF download and `{curp_id}.pdf` naming works
- ✅ **Milestone 4**: Database orchestrator manages complete workflow
- ✅ **Milestone 5**: End-to-end test: CURP input → PDF delivery

Follow this sequential plan and you'll have a complete, production-ready CURP-to-PDF automation system!

Start with Phase 1 (completing Outlook creation) and work through each phase systematically. Each phase builds on the previous one, ensuring a solid foundation.