# ChromeSync Development Guide: Complete Checklist

## Project Overview

ChromeSync is a Python-based tool that synchronizes browsing data (passwords, history, and bookmarks) from Google Chrome to Zen Browser on Windows 11. This checklist outlines the development process in chronological order.

## Phase 1: Environment Setup and Initial Planning

### Development Environment Configuration

- ☑ ~~Install latest stable Python version~~
- ☑ ~~Set up a virtual environment for dependency isolation~~
- ☑ ~~Install required Python libraries:~~
  - ☑ ~~psutil (for process monitoring)~~
  - ☑ ~~sqlite3 (for database interactions)~~
  - ☑ ~~pyautogui or selenium (for GUI automation)~~
  - ☑ ~~json (for handling JSON-formatted data)~~
  - ☑ ~~pandas (for data manipulation)~~
  - ☑ ~~cryptography (for security operations)~~
  - ☑ ~~pywin32 (for Windows-specific operations)~~
  - ☑ ~~tkinter or PyQt5 (for GUI development)~~
  - ☑ ~~pillow (for image handling in the GUI)~~

### Project Structure Setup

- ☑ ~~Create the main project directory~~
- ☑ ~~Set up the following directory structure:~~
  - ☑ ~~`/src` - Main source code directory~~
  - ☑ ~~`/src/core` - Core functionality modules~~
  - ☑ ~~`/src/utils` - Utility and helper functions~~
  - ☑ ~~`/src/gui` - GUI components~~
  - ☑ ~~`/src/config` - Configuration handling~~
  - ☑ ~~`/src/security` - Security-related modules~~

    - [x] ~~`/tests` - Test files~~
    - [x] ~~`/docs` - Documentation~~
    - [x] ~~`/resources` - Static resources~~
- [x] ~~Create a README.md with project description~~
- [x] ~~Create a requirements.txt file~~
- [x] ~~Set up a version control system (git)~~
- [x] ~~Create .gitignore file~~

### Configuration Management

- [x] ~~Create a configuration module for:~~
    - [x] ~~Browser paths (Chrome and Zen Browser)~~
    - [x] ~~User profile information~~
    - [x] ~~Sync preferences (what to sync)~~
    - [x] ~~Temporary file paths~~
    - [x] ~~Security settings~~
    - [x] ~~Synchronization schedule and cadence options~~
    - [x] ~~GUI preferences~~
    - [x] ~~Logging verbosity control~~
- [x] ~~Create configuration file templates~~
- [x] ~~Implement configuration validation~~

## Phase 2: Chrome Process Monitoring

### Chrome Process Detection

- [ ] Implement background service using psutil
- [ ] Create an efficient polling mechanism
- [ ] Add logging for process detection events
- [ ] Design interface for passing monitoring events to GUI components

### Service Management

- [ ] Implement auto-startup capability (Windows Task Scheduler integration)
- [ ] Create service control functions (start, stop, status)
- [ ] Add error handling and recovery mechanisms
- [ ] Implement graceful shutdown procedures
- [ ] Create API hooks for GUI-based service control

## Phase 3: Chrome Data Extraction

### Password Extraction Module

- [ ] Implement GUI automation for password export to CSV
  - [ ] Create navigation functions using pyautogui or selenium
  - [ ] Add automatic export to secure temporary location
- [ ] Implement direct database access method (alternative)
  - [ ] Create functions to access Chrome's Login Data SQLite database
  - [ ] Implement decryption using Windows Data Protection API
- [ ] Add fallback mechanism between methods
- [ ] Add progress reporting interface for GUI integration

### Bookmark Extraction Module

- [ ] Implement direct access to Chrome's Bookmarks JSON file
- [ ] Create data parser for hierarchical bookmark structure
- [ ] Generate Zen Browser compatible format (HTML)
- [ ] Add progress tracking and status reporting

### History Extraction Module

- [ ] Implement SQLite database access to Chrome's History file
- [ ] Create queries for relevant browsing history data
- [ ] Add timestamp conversion functionality
- [ ] Structure data for Zen Browser compatibility
- [ ] Implement cancellation points for user interruption

## Phase 4: Zen Browser Import Implementation

### Browser Profile Detection

- [ ] Create function to identify active Zen Browser profiles
- [ ] Parse the about:profiles page
- [ ] Validate profile paths and access permissions
- [ ] Add user selection for multi-profile scenarios

### Password Import Module

- [ ] Implement GUI automation for CSV import into Zen Browser
- [ ] Add secure handling of temporary password files
- [ ] Implement file existence and format verification
- [ ] Add immediate secure deletion of temporary files

- [ ] Develop detailed progress and error reporting

### Bookmark Import Module

- [ ] Implement GUI automation for HTML bookmark import
- [ ] Add verification for successful import
- [ ] Create error handling for import failures
- [ ] Add operation outcome feedback mechanisms

### History Import Module

- [ ] Research and implement the best approach:
  - [ ] Option 1: Profile migration method
  - [ ] Option 2: Direct database manipulation (places.sqlite)
  - [ ] Option 3: Firefox Sync integration (if available)
- [ ] Implement the most reliable method with fallbacks
- [ ] Add detailed progress tracking for long-running operations

## Phase 5: Security Implementation

### Secure Data Handling

- [ ] Implement memory-safe data handling practices
- [ ] Minimize persistence of sensitive data (especially passwords)
- [ ] Add secure file operations with immediate cleanup
- [ ] Implement encryption for any temporary storage
- [ ] Design secure handling of configuration in GUI context

### User Authentication

- [ ] Add Windows user authentication for sensitive operations
- [ ] Implement privilege checking for database access
- [ ] Create secure handling of decryption keys
- [ ] Develop GUI authentication for sensitive operations

### Security Audit and Hardening

- [ ] Review all code paths for security vulnerabilities
- [ ] Implement defense-in-depth strategies
- [ ] Add tamper detection for critical files
- [ ] Document security considerations for users

- [ ] Audit GUI security to prevent information leakage

# Phase 6: Error Handling and Logging

## Comprehensive Error Handling

- [ ] Implement exception handling for all critical operations
- [ ] Create graceful degradation strategies (partial sync if full sync fails)
- [ ] Add retry mechanisms with exponential backoff
- [ ] Implement self-recovery procedures
- [ ] Develop user-friendly error presentation in GUI

## Logging System

- [ ] Create detailed logging with different verbosity levels
- [ ] Implement secure logging (no sensitive data in logs)
- [ ] Add log rotation and management
- [ ] Create user-friendly error messages
- [ ] Add log viewer component to GUI for troubleshooting

# Phase 7: GUI Implementation

## Main Application Window

- [ ] Design and implement the primary application window
- [ ] Create a clean, intuitive interface with modern design
- [ ] Implement responsive layout for various screen sizes
- [ ] Add application status indicators and controls

## Configuration Interface

- [ ] Develop a comprehensive settings panel with sections for:
    - [ ] Browser paths and profiles
    - [ ] Data types to synchronize
    - [ ] Synchronization schedule options
    - [ ] Security settings
    - [ ] Advanced options
- [ ] Implement validation for user inputs
- [ ] Add configuration import/export functionality
- [ ] Create preset configurations for common use cases

### Synchronization Controls

- [ ] Implement manual synchronization triggers:
  - [ ] Full synchronization option
  - [ ] Selective synchronization (passwords only, bookmarks only, etc.)
  - [ ] Scheduled synchronization configuration
- [ ] Add start/stop/pause controls for background monitoring
- [ ] Create sync history view for past operations

### Status and Feedback Components

- [ ] Develop real-time status display for ongoing operations
- [ ] Implement progress indicators for multi-step processes
- [ ] Create visual notifications for completed operations
- [ ] Add detailed error reporting with troubleshooting suggestions
- [ ] Implement status logging with filtering options

### System Tray Integration

- [ ] Create Windows system tray icon and context menu
- [ ] Implement quick access to common functions
- [ ] Add status indicators in the tray icon
- [ ] Develop minimize-to-tray functionality
- [ ] Implement startup behavior options

## Phase 8: Testing and Validation

### Unit Testing

- [ ] Develop comprehensive tests for each module
- [ ] Implement mock objects for browser interactions
- [ ] Create test data fixtures for consistent testing
- [ ] Add GUI component testing

### Integration Testing

- [ ] Test full synchronization workflow
- [ ] Verify data integrity after synchronization
- [ ] Validate security measures
- [ ] Test error recovery mechanisms
- [ ] Test GUI interaction with core functionality

### Performance Testing

- [ ] Measure and optimize synchronization speed
- [ ] Monitor resource usage during various operations
- [ ] Benchmark different implementation approaches
- [ ] Optimize database queries and file operations
- [ ] Test GUI responsiveness during intensive operations

### Usability Testing

- [ ] Conduct user testing of the GUI interface
- [ ] Gather feedback on workflow and design
- [ ] Test with users of varying technical skill levels
- [ ] Implement improvements based on feedback

## Phase 9: Packaging and Deployment

### Installer Creation

- [ ] Package the application with all dependencies
- [ ] Create Windows installer with proper permission requests
- [ ] Implement auto-update capability
- [ ] Add first-run configuration wizard
- [ ] Add GUI customization options during installation

### Documentation

- [ ] Create comprehensive user documentation
- [ ] Add developer documentation for future maintenance
- [ ] Document security considerations and best practices
- [ ] Create troubleshooting guide
- [ ] Add in-app help system and tooltips

## Phase 10: Maintenance and Improvement Plan

### Ongoing Compatibility Monitoring

- [ ] Plan for Chrome and Zen Browser updates
- [ ] Implement version detection and compatibility checks
- [ ] Create update mechanism for the synchronization tool

- [ ] Add browser version tracking and notifications

### Feature Expansion

- [ ] Plan for additional data types (form data, cookies, etc.)
- [ ] Consider bidirectional synchronization
- [ ] Explore additional browser support
- [ ] Add user feedback mechanism for feature requests

## Risk Management

### Identify and Address Key Challenges

- [ ] Create mitigation strategies for secure password handling
- [ ] Implement flexible parsers to handle browser updates
- [ ] Develop multiple approaches for complex data synchronization
- [ ] Optimize performance for large data sets
- [ ] Ensure GUI responsiveness during intensive operations

## Version Release Planning

### Alpha Release

- [ ] Core functionality (Chrome extraction + Zen import)
- [ ] Basic CLI interface
- [ ] Essential error handling
- [ ] Limited testing

### Beta Release

- [ ] Complete GUI implementation
- [ ] Advanced security features
- [ ] Comprehensive error handling and logging
- [ ] Extensive testing
- [ ] Limited user feedback incorporation

### Version 1.0 Release

- [ ] All planned features complete
- [ ] Comprehensive testing passed

- [ ] Documentation complete
- [ ] Installer and deployment package ready
- [ ] User feedback incorporated

## Weekly Progress Tracking

| Week | Planned Tasks | Status | Notes |
|---|---|---|---|
| 1 | Environment Setup | Completed | |
| 2 | Chrome Process Monitoring | Inprogress | |
| 3-4 | Chrome Data Extraction | | |
| 5-6 | Zen Browser Import | | |
| 7 | Security Implementation | | |
| 8 | Error Handling and Logging | | |
| 9-11 | GUI Implementation | | |
| 12-13 | Testing and Validation | | |
| 14 | Packaging and Deployment | | |
| 14.5 | Maintenance Planning | | |