## 1. Problem Understanding

The objective of this assignment is to extract publicly visible data from LinkedIn company pages and individual LinkedIn profile URLs. Since LinkedIn is a dynamic and protected platform, the challenge is not just extracting data, but doing so in a controlled and reliable way without aggressive scraping.

---

## 2. Technology Choice

I used **Node.js with Playwright** for this assignment.

LinkedIn pages rely heavily on JavaScript and load most content dynamically. Browser automation using Playwright allows the scraper to behave like a real user, which makes it more reliable than direct HTTP-based scraping.

---

## 3. Authentication Handling

LinkedIn requires authentication to access most profile and company information.

The scraper uses a session-based login approach:

- User logs in manually on the first run
- Session cookies are saved locally
- Cookies are reused in subsequent runs
- If cookies expire or are missing, login is performed again

This avoids repeated login attempts and reduces the risk of account blocking.

---

## 4. Handling Infinite Scrolling and Pagination

Employee listings on LinkedIn company pages are loaded using infinite scrolling.

To handle this:

- The scraper scrolls the page gradually
- Waits between scrolls to allow data to load
- Extracts employee cards after scrolling completes
- For sample output, extraction is limited to the first 10 employees

---

## 5. Rate Limiting and Bot Detection Awareness

To minimize detection:

- No parallel requests are made
- Delays are added between actions
- Pages are navigated sequentially
- Only required pages are visited

The goal is to keep the scraping behavior close to normal human usage.

---

## 6. Data Extraction Strategy

Data is extracted from rendered HTML using Playwright selectors.

**Company Page:**

- Employee name
- Current title
- Profile URL
- Location (if available)

**Profile Page:**

- Name
- Headline
- Location
- Experience
- Education
- Skills (if visible)
- Connections count (if shown)

If any field is not available, it is returned as `null` instead of failing the process.

---

## 7. Error Handling

The scraper includes basic error handling for:

- Missing elements
- Slow page loads
- Private or restricted sections

- Invalid or expired cookies

Errors are logged, and the scraper continues execution wherever possible.

---

## 8. Scalability Considerations

To scale this solution for large volumes (e.g., 10,000 profiles per day), the following approach can be used:

- Job queue system (BullMQ or similar)
- Multiple worker processes
- Account rotation
- Controlled concurrency per account

---

## 9. Limitations

- LinkedIn UI changes can break selectors
- CAPTCHA may appear if scraping is aggressive
- Only publicly visible data is extracted
- Requires a valid LinkedIn account

---

## 10. Conclusion

This solution focuses on reliability, clarity, and real-world constraints rather than aggressive data extraction. The design allows safe scraping at small scale and can be extended for larger volumes if required.