Internship Daily Work Documentation (Day 1 - Day 30)

Project: ID Card Content Extraction using OCR & NLP
Internship at Turtil (May-June 2025)

High-Level 6-Week Work Plan Overview

Week	Focus Area	Deliverables
Week 1	📦 Dataset, Regex Design, Project Setup	Synthetic Dataset + Regex + Project Skeleton
Week 2	🔍 Offline OCR + Baseline Extraction	Working OCR → Basic Text Extraction
Week 3	🧠 Entity Extraction with ML (NER)	Trained spaCy/Sklearn NER model
Week 4	🧪 Integration, API, Docker, Testing	FastAPI app + Docker + Test cases + Docs

📅 **WEEKLY PLAN (Deadline-Oriented)**

✅ **WEEK 1: Dataset Creation + Regex Design + Project Skeleton**

📅 Goal: Create dataset, design regex patterns, prepare base project folders

✅ **WEEK 2: Offline OCR + Baseline Field Extraction**

📅 Goal: Use Tesseract OCR to extract raw text → regex matcher baseline

✅ **WEEK 3: ML-Based Entity Extraction (NER with spaCy or sklearn)**

📅 Goal: Train spaCy/Sklearn model to predict field values from text

✅ **WEEK 4: Integration, FastAPI, Docker, Testing & Deployment**

📅 Goal: Build API + Test + Package in Docker for deployment

✅ **WEEK 5-6: Working on Accuracy Improvements**

📅 Goal: Improve Accuracy+Re_deploy

Documentation

📘 ID Card Content Extractor Internship Log

✅ Day 1-2: Dataset Setup + Project Initialization (Date: 19-20th May)

🎯 Objective:

- Set up initial folder structure and tools
- Create synthetic ID card dataset with labels
- Prepare regex configuration for field extraction

📁 Tasks Completed:

- ✔ Created project folder structure:
- ✔ Generated **10 synthetic student ID cards** using Canva (names, roll numbers, branches, colleges, validity).
- ✔ Created **corresponding JSON label files** for each image with structure:
- {
- "user_id": "stu_001",
- "extracted_fields": {
- "name": "Anjali Sharma",
- "college": "RGMCET Nandyal",
- "roll_number": "21RGME1032",
- "branch": "Mechanical Engineering",
- "valid_upto": "2025"
- }
- ✔ Defined initial **regex rules and confidence weights** in config.json
- ✔ Wrote a Python utility to validate and preview all JSON label files

📌 Output:

- Dataset ready with images + labels
- config.json containing field rules and regex patterns

✅ Day 3-4: Tesseract OCR + Regex Field Extractor (Date:20-21st May)

🎯 Objective:

- Setup offline OCR using Tesseract
- Extract raw text from images
- Use regex patterns to match fields from OCR output

🛠 Tasks Completed:

✔ Installed **Tesseract OCR engine** and integrated it with Python (pytesseract)

✔ Wrote ocr.py to:

- Read ID card image
- Convert to grayscale using OpenCV
- Extract raw text using Tesseract

✔ Wrote regex_extractor.py to:

- Load regex patterns from config.json
- Match fields like roll_number, valid_upto
- Calculate confidence score and track missing fields

✔ Created and ran test_pipeline.py:

- Executed OCR + field extraction
- Printed both raw text and final structured output

✔ Handled platform-specific Tesseract path issue using:

- pytesseract.pytesseract.tesseract_cmd = r"C:\\Program Files\\Tesseract-OCR\\tesseract.exe"

🔍 Sample OCR Output (Example):

OCR Output: NIT Trichy

Akhil Varma HTNo_ NITT21ECEI001

Branch Electronics and Communication   Valid 2026 upto

📤 Sample Structured Output:

Extracted Fields:

```
{
 "extracted_fields": {
   "roll_number": null,
   "valid_upto": "2026",
   "name": null,
   "college": null,
   "branch": null
 },
 "confidence_score": 0.03,
 "missing_fields": [
   "roll_number",
   "name",
   "college",
   "branch"
 ],
 "status": "partial_success"
}
```

✅ Day 5: spaCy NER Training + Smart Entity Extraction (Date: 22st May)

🎯 Objective:

Train and integrate a custom **spaCy Named Entity Recognition (NER)** model to extract entities like name, college, and branch from OCR output, and integrate fallback logic to ensure 100% structured output.

🗂️ Tasks Completed:

✅ Installed and configured spaCy and en_core_web_sm

✅ Created utility to generate **NER training data** from labeled OCR + JSON files

✅ Trained a custom **NER model** with 22 labeled samples using spaCy.blank("en")

✅ Saved the trained model to ner_model/

✅ Integrated the trained model into pipeline via ner_extractor.py

✅ Wrote a fallback extractor (fallback_extractor.py) using **keyword matching** for college and branch (a custom data)

✅ Cleaned and normalized OCR text before passing to NER for better accuracy

✅ Merged predictions from:

- Regex
- NER
- Fallback extractor

✅ Finalized output format to avoid null by defaulting to "Unknown" if all extraction fails

✅ Sample Final Output:

```
{
  "roll_number": "22ANN3362",
  "valid_upto": "2025",
  "name": "John Farmer",
  "college": "Anna University",
  "branch": "Mechanical Engineering"
}
```

🧠 **Key Learnings:**

- Understood how spaCy NER pipelines work and how to fine-tune them
- Learned how to blend ML, pattern-based, and keyword-based systems to achieve 100% extraction coverage
- Designed for robustness and consistency in production

Final Outcome: (Accuracy)

- Field-level accuracy for 22 images and it's corresponding json

✅ Total Fields: 110 ✅ Correct Fields: 86 🎯 **Field-Level Accuracy: 78.18% (Initial Accuracy)**

✅ **Day 6: (Date: 22-23nd May)**

✅ **Tasks Completed:**

- Installed required libraries:
- pip install fastapi uvicorn python-multipart
-
- Created FastAPI app with endpoint /extract
- Integrated the core pipeline (ocr.py, regex_extractor.py, ner_extractor.py)
- Used **Pydantic** models to validate input and output formats
- Auto-generated Swagger UI at:

👉 http://localhost:8000/docs

- Created and tested a Python script test_api.py to send base64 image to the API and get predictions
- Validated JSON response contains predicted fields (name, roll_number, etc.)

🧠 **Outcome:**

- App now exposes a professional, structured **microservice endpoint**
- Users can POST a base64 image to receive **OCR + NER + Fallback** predictions

✅ 📄 Day-7 Documentation – Dockerizing the Microservice(23rd May)

🔹 Goal:

To containerize the entire FastAPI microservice into a portable Docker image that can run fully offline.

✅ Tasks Completed:

- Installed Docker Desktop (x86_64 version for Windows)
- Validated installation using:
- docker run hello-world
- Created a Dockerfile with:
  - Base image: python:3.10-slim
  - Installed system deps: tesseract-ocr, libgl1
  - Installed Python deps from requirements.txt
  - Downloaded spaCy model
  - Exposed port 8000
  - Launched app using uvicorn
- Built Docker image:
- docker build -t idcard-extractor .
- Ran container and verified:
- docker run -p 8000:8000 idcard-extractor
- Accessed microservice from browser at:

👉 http://localhost:8000/docs

- Confirmed test_api.py worked even when API was running inside Docker

🧠 Outcome:

- Entire project is now portable and isolated
- Can be deployed, shared, or tested on any system without needing local setup
- Runs fully offline as per project constraints

📘 Day-8-9 Documentation – Minimal Modern UI + API Integration(26-27th May 2025)

✅ Tasks Completed Today:

Task	Description
🧠 Integrated API with Frontend	Built a basic web UI to test /extract endpoint of FastAPI
🖼️ Image Upload + Preview	Added file upload using HTML5 with image preview functionality
📤 Base64 Conversion	Handled client-side conversion of uploaded image to base64
🧾 JSON Display	Displayed extracted JSON response neatly under the image
💻 Code Editor Style Output	Used highlight.js + Tailwind CSS to show JSON with syntax highlighting
⚡ FastAPI Integration	Hooked up frontend to call FastAPI /extract endpoint using fetch()
🎯 Output Handling	Rendered response dynamically, clean, scrollable and copyable

💡 Key Features Implemented:

- HTML5 + Tailwind CSS for layout
- highlight.js for syntax-highlighted code output
- Replaced form submission with fetch() (AJAX) for smoother UX
- Automatically previews the uploaded image before submission
- JSON is rendered in <pre><code> like code editors

📸 UI Workflow:

1. **User uploads image**
2. **Preview appears**

3. **Button click triggers fetch() POST**
4. **FastAPI returns structured JSON**
5. **JSON appears beautifully below the image**

📘 Day-10-11 Documentation – UI Enhancements & Responsiveness(27-28th May 2025)

✅ Tasks Completed Today:

Task	Description
🎨 Upgraded upload.html	Improved UI using **Tailwind CSS** to be modern, responsive, and mobile-friendly
🖼️ Image Preview	Added real-time preview for uploaded image
🔄 Loading Spinner	Integrated a spinner to show extraction in progress
🧾 JSON Result Display	Styled extracted JSON in a clean, code-editor-like box
📋 Copy to Clipboard	Added a button to copy the JSON result easily
💾 Download Button	Added functionality to download extracted JSON as .json file
📱 Responsive Layout	Optimized for different screen sizes (desktop, mobile, tablet)

🧠 Technical Features Added:

- Used FileReader to preview image and convert to Base64
- Used fetch() to POST the image to /extract endpoint
- Dynamically injected server response in a <pre><code> box
- CSS animations for smooth UX using Tailwind's utilities
- Improved file structure and front-end JavaScript logic

🚀 Preview

Users can now:

- Upload an image
- See it previewed instantly
- Click extract and watch the loader
- See results in a neatly formatted code editor style
- Copy or download results easily

📘 Day-12 Documentation (28th May 2025)

✅ Work Done

- Successfully integrated UI-based image upload to FastAPI.
- Reorganized and cleaned backend architecture for maintainability.
- Used StaticFiles and Jinja2Templates to serve HTML UI via FastAPI.
- Added TailwindCDN styling for better visual layout and cleaner code UI.
- Implemented JSON preview with syntax highlighting using highlight.js.
- Ensured **offline compatibility** by downloading necessary frontend dependencies (Tailwind CSS, highlight.js).
- Dockerized the complete application and fixed static path errors.
- Finalized successful deployment to **Render.com**.
- Tested deployment via browser and confirmed working UI and functionality.

📘 Day-13-15 Documentation (29-30th May 2025)

✅ Work Done Today

- Refined NER output to a **new enhanced response format**:

```
{
  "user_id": "stu_1234",
  "extracted_fields": { ... },
  "confidence_score": 0.91,
  "missing_fields": ["branch"],
  "status": "partial_success"
}
```

- Defined status based on the number of fields extracted:
    - success – all fields present
    - partial_success – some fields missing
    - failure – all fields missing/unrelated image
- Added confidence_score calculation using:

1 - (missing / total_fields)

- Retested entire pipeline with test images.
- Clarified FastAPI's behavior with response_model vs raw dictionary return.

✅ 📄 Day-11 Documentation – JSON Structuring & Deployment (2nd -3rd June 2025)

🔹 **Goal:**

Enhance API output and structure response to include metadata and deployment integration.

✅ **Tasks Completed:**

- Integrated advanced JSON response structure with:
    - user_id
    - confidence_score
    - missing_fields
    - status
- Defined status logic:
    - success (all fields)
    - partial_success (some missing)
    - failure (none extracted)
- Refactored FastAPI /extract endpoint accordingly.
- Conducted local testing via Swagger UI and test_api.py.
- Rebuilt Docker image, pushed code to GitHub.
- Re-deployed enhanced application on Render.

🧠 **Outcome:**

- Structured API now ready for production use.
- Easier to consume and debug based on status logic.
- Cloud-hosted microservice accessible publicly.

✅ Demo Link: idcard-extractor.com

✅ 📄 Day-12 Documentation – Model Accuracy Boost (4th June 2025)

🔹 **Goal:**

Improve model generalization by training on diverse field values.

✅ **Tasks Completed:**

- Improved model accuracy from **82% to 90%**.
- Added value variants:
    - 'Computer Science' → 'CSE'
    - 'Roll No.', 'HTNO', 'Card No.'
- Retrained model on updated dataset.

🧠 **Outcome:**

- Field-level NER extraction became more robust.
- Reduced misclassifications due to unseen patterns.

✅ 📄 Day-13 Documentation – Accuracy Optimization (5th June 2025)

🔹 **Goal:**

Continue enhancing field-level prediction logic.

✅ **Tasks Completed:**

- Focused on refining normalization and regex patterns.
- Validated results with augmented samples.

🧠 **Outcome:**

- Model predictions began stabilizing across edge cases.

✅ 📄 Day-14 Documentation – Field-Specific Improvements (6th June 2025)

- 🔹 **Goal:**

Fix inconsistencies in BRANCH and ROLL_NUMBER field extraction.

✅ **Tasks Completed:**

- Tuned regex + fallback for BRANCH and ROLL_NUMBER.
- Ran repeated validations on real ID card images.

🧠 **Outcome:**

- Better field isolation and recall in predictions.

✅ 📄 Day-15 Documentation – Label Merging & Retraining (9th June 2025)

🔹 **Goal:**

Simplify data management and improve training consistency.

✅ **Tasks Completed:**

- Merged multiple label files into one merge_labels.json.
- Updated augmentation script to use merged labels.
- Retrained NER with augmented dataset.
- Achieved **85.29%** accuracy using evaluate_accuracy.py.

🧠 **Outcome:**

- Reduced duplication and noise in labels.
- Centralized label control improved training flow.

✅ 📄 Day-16 Documentation – Validation and Re-Testing (10th June 2025)

🔹 **Goal:**

Ensure merged label structure holds across training/evaluation.

✅ **Tasks Completed:**

- Revalidated training pipeline using merged data.
- Measured improvement in:
    - Name
    - Branch
    - Valid Upto fields

🧠 **Outcome:**

- Confirmed accuracy uplift from merged training set.

✅ 📄 Day-17 Documentation – README Creation (11th June 2025)

🔹 **Goal:**

Document the project comprehensively for public GitHub visibility.

✅ **Tasks Completed:**

- Created structured README.md file:
    - Introduction
    - Features
    - API usage
    - Setup instructions

🧠 **Outcome:**

- Project is now self-documented and easy for others to understand/setup.

✅ 📄 Day-18 Documentation – Presentation (12th June 2025)

🔹 **Goal:**

Summarize project in a visual, mentor-friendly format.

✅ **Tasks Completed:**

- Created presentation (PPT) covering:
    - Problem statement
    - Architecture
    - Pipeline steps
    - Accuracy benchmarks

🧠 **Outcome:**

- Ready for final mentor/demo presentation.

✅ 📄 Day-19 Documentation – Setup Guide (13th June 2025)

🔹 **Goal:**

Write a detailed setup guide for offline and online deployment.

✅ **Tasks Completed:**

- Documented:
  - Local installation
  - Docker setup
  - API testing steps
- Verified offline usage constraints.

🧠 **Outcome:**

- Enables any user to run and test the app independently.

✅ 📄 Day-20 to Day-22 Documentation – Field Accuracy Boost (16th June 2025)

🔹 **Goal:**

Enhance accuracy in Branch and College fields.

✅ **Tasks Completed:**

- Integrated normalization for:
  - Branch
  - Name
  - College
  - Roll Number
- Improved fallback logic and unseen value handling.
- Final accuracy improved to **91.18%**.

🧠 **Outcome:**

- System now handles both common and rare ID formats well.

✅ 📄 Day-23 Documentation – Evaluation Metrics (17th June 2025)

🔹 **Goal:**

Validate improvements via evaluate_accuracy.py.

✅ **Tasks Completed:**

- Ran evaluations on full test dataset (510 fields).
- Verified:
  - Valid Upto: 97%
  - Branch: 95%
  - Roll Number: 88%
  - Name/College: 84–86%

🧠 **Outcome:**

- Final performance benchmarks established.

✅ 📄 Day-24 Documentation – Final Review (18th June 2025)

🔹 **Goal:**

Ensure all project components work together seamlessly.

✅ **Tasks Completed:**

- Cross-verified model, pipeline, frontend, and API.
- Confirmed working from end-to-end.

🧠 **Outcome:**

- Final version validated for submission/demo.

✅ 📄 Day-25 Documentation – Deployment & Branching (19th June 2025)

🔹 **Goal:**

Push final stable version with 91% accuracy to GitHub.

✅ **Tasks Completed:**

- Created v2 branch.
- Pushed all normalized + optimized code.
- Re-deployed to Render.

🧠 **Outcome:**

- Production-grade release with version control.

✅ 📄 Day-26 Documentation – Accuracy Table (20th June 2025)

🔹 **Goal:**

Summarize accuracy progression across all fields.

✅ **Tasks Completed:**

Field	**Initial**	**Final**
Name	80%	86%
College	78%	84%
Roll Number	62%	88%
Branch	84%	95%
Valid_Upto	91%	97%
Overall	**80%**	**91%**

🧠 **Outcome:**

- Quantified the impact of all optimization efforts.
- Achieved 91% as the **(Final Accuracy).**

✅ 📄 Day-26 Documentation – test_api.py Validation (23th June 2025)

🔹 **Goal:**

Validate API response through external test scripts.

✅ **Tasks Completed:**

- Ran test_api.py to verify:
    - JSON structure
    - Normalized values
    - Proper API status codes

🧠 **Outcome:**

- CLI-based testing confirmed backend logic robustness.

✅ 📄 Day-27 Documentation – Final Cleanup (24th June 2025)

🔹 **Goal:**

Eliminate edge-case bugs and finalize output.

✅ **Tasks Completed:**

- Fixed .strip() on None type bug.
- Unified normalization logic across:
    - CLI
    - FastAPI
    - Evaluation pipeline

🧠 **Outcome:**

- Ready for stable use across all modes.

✅ 📄 Day-28 Documentation – Documentation Finalization (25th June 2025)

🔹 **Goal:**

Consolidate and polish all documentation deliverables.

- ✅ **Tasks Completed:**
- Completed README, Setup Guide, and Presentation updates.
- Added version logs for v2.

🧠 **Outcome:**

- Project artifacts now presentable and ready for review.

✅ 📄 Day-29-30 Documentation – Internship Wrap-Up (26th- 27th June 2025)

🔹 **Goal:**

Conclude internship with final review and submission.

✅ **Tasks Completed:**

- Final accuracy confirmed: **91.18%**
- Demo prepared.
- Code, documents, and slides submitted.

🧠 **Outcome:**

- Successfully built and deployed an offline-capable OCR + NLP microservice for ID card extraction and submitted.

ThankYou Turtil and the Entire Team