The report should be brief and give me an understanding of the following:

• Which significant choices you have had to make during development, what the options were, and why you made them.

We decided to use pandoc to convert markdown files to PDF due to its ability to include images like the NTNU logo and signatures. Though slower than alternatives like pdfkit, pandoc's simplicity made it our preferred choice.

• How the system works and how it is structured.

The project comprises a frontend and a backend. The frontend allows file uploads, which are then processed by the backend. The backend extracts information from CSV files, inserts them into markdown templates, and converts them to PDF using pandoc.

• An overview of the code (regular documentation should be in code).

Backend part of the code:

- Backend:
  - Filling Templates: Data from a source fills templates, creating Markdown files.
  - Converting Markdown to PDF: Pandoc converts Markdown files to PDFs.
  - Error Handling: Non-zero exit status from pandoc results in error handling.
- Frontend:
  - Directory Setup: Defines and creates directories for file handling.
  - File Upload Endpoint: Defines an endpoint for file uploads.
  - File Download Endpoint: Defines an endpoint for file downloads.

• How to use the system.

Installation: Clone repository, navigate to project directory, and install dependencies.

Building Docker Image: Build the Docker image for the backend server using provided Dockerfile.

Running Locally: Start the server using main.py and access endpoints through the browser or Postman. Backend server must be running for file processing. In the upload.py there is a different URL for the localhost than the railway server.

Accessing Hosted System: Frontend and backend servers are hosted on Railway. Access them at the provided URLs.

• How to set up a system like this.

Break down the development into manageable tasks. Begin with setting up an API to handle file processing and a frontend to interact with the API.

- **How to replace the file processing with another use case.**

    - Identify New Use Case: Determine the new system functionality.
    - Modify Backend: Update backend code to handle the new functionality.
    - Update API Endpoints: Modify API endpoints accordingly.
    - Modify Frontend: Update frontend to interact with new API endpoints.
    - Testing: Test the system to ensure proper functionality with the new use case.