BUILDTECH

A CONSTRUCTION MANAGMENT SYSTEM

DEVELOPER GUIDE

Lena Nemah Dabbah

209140268

2025

BUILDTECH

Table Of Contents

Welcome To Our Project

BuildTech is a comprehensive construction project management system designed specifically to streamline the management of construction projects. It is intended for managers and administrators who are responsible for overseeing and tracking the progress of construction projects from start to finish and maintaining safety which is the important part I want to emphasize and present in this document.

Our Service

The BuildTech system consists of multiple technologies and over 30 pages. The system is modular and organized in different folders according to functionality.

BuildTech is a web-based construction project management system designed primarily for site managers and administrative staff.

The core service of the system is:

Maintaining safety on construction sites by analyzing images using a machine learning model.

Glossary

Term	Description
BuildTech	The name of the system – a construction project management and safety monitoring platform.
Landing Page	The first page of the website, accessible to all users. Contains navigation and a contact form.
Manager Login	A secure area accessible only to authorized managers or administrators.
Navbar	Navigation bar located on the right side of the screen after login, used to move between system pages.
Dashboard	The home view after login, showing summaries like project lists, task status, maps, and charts
Project Details Page	A page showing full project info, including its location, attached files, and image upload
Not Analyzed	A status for uploaded images that have not yet been reviewed by the safety detection system.
Safety Analysis	An automated process that checks images using a Machine Learning model to detect safety gear.
ML (Machine Learning)	A method that enables the system to analyze images and detect whether safety measures are followed.
PDF Report	A file generated by the system after analysis, showing which images include safety issues.
PHPMailer	A PHP library used to send the email reports with attachments (like PDF).

Term	Description
XAMPP	A local development environment that includes Apache, PHP, and MySQL for testing the system.
PhpMyAdmin	A tool used to manage the MySQL database through a web interface.
API	Application Programming Interface – used to allow different parts of the system (or external tools) to communicate.
Cron Job	A scheduled task (every 1 hour) that triggers the safety image analysis script.
Frontend	The part of the application that users interact with directly. It includes the user interface and is typically built using HTML, CSS, and JavaScript frameworks like React or Vue.
Backend	The server-side part of the application that handles business logic, database interactions, authentication, and other core functionality. It's usually written in languages like PHP, Python, Node.js, etc.
Database	A structured collection of data that the backend interacts with to store, retrieve, and manage application data. Common databases include MySQL, PostgreSQL, MongoDB, etc.

Project Architecture

Backend

- PHP - The core backend logic and server-side processing are primarily written in PHP. It handles routing, database interactions, authentication, session management, and rendering dynamic HTML templates. PHP also acts as the controller layer that connects frontend views to backend data and logic.
- Python - Python is used in a limited number of scripts for tasks such as machine learning and image analysis. These scripts are triggered by PHP processes and support advanced features like safety evaluation based on image data.

Frontend

- HTML
  Defines the structure and layout of all web pages. Pages are generated dynamically with embedded PHP variables and content.
- CSS
  Manages the styling and layout of the user interface, defining visual components like colors, fonts, spacing, and responsive behavior.
- JavaScript
  Adds interactivity and real-time communication with the backend. Responsibilities include:
  - Populating HTML with dynamic data.
  - Handling form submissions via AJAX without full page reloads.
  - Displaying modals and tooltips.
  - Updating project statuses and financial info live.
  - Uploading/deleting files with user feedback.
  - Formatting numbers and currencies.
  - Rendering charts to display project data.
  - Reloading specific sections of the page after actions.

Database

- **MySQL**
  Used as the primary relational database to store structured data including:
  - **User accounts**
  - **Projects and tasks**
  - **Uploaded images**
  - **System configurations**
- **The database is managed using phpMyAdmin within the XAMPP environment, providing an easy-to-use interface for browsing and editing data during development and testing.**

Machine Learning Model

This section describes the machine learning component used for image analysis, including dataset source, preprocessing, model architecture, training, and integration into the system.

Dataset

The dataset used for training the machine learning model is the Construction Site Safety Image Dataset, available on Kaggle at: https://www.kaggle.com/datasets/snehilsanyal/construction-site-safety-image-dataset-roboflow

This dataset contains labeled images of construction workers and equipment, with annotations indicating the presence or absence of safety equipment.

Original Dataset Statistics from Kaggle:

Folder	Purpose	Image Count
train/	Training the model	2,605 images
valid/	Validation during training	114 images
test/	Final evaluation of the model	997 images

Annotations:

- Annotation format: YOLO v5 format (.txt files)
- Each image has an associated label file containing:
  - Class index (0-9 based on the class list)
  - Bounding box coordinates (normalized)

Classes in the Dataset:

Class Index	Class Name
0	Hardhat
1	Mask
2	NO-Hardhat
3	NO-Mask
4	NO-Safety Vest
5	Person
6	Safety Cone
7	Safety Vest
8	machinery
9	vehicle

This dataset provides a comprehensive foundation for training models aimed at enhancing safety on construction sites through the detection of personal protective equipment (PPE) and related safety measures.

Dataset Folder Structure

The dataset used for training, validation, and testing follows the folder structure below. This organization is required for YOLOv8 to correctly locate the images and their corresponding annotation files.

Dataset Directory Tree:

```
css-data/
├── README.dataset.txt
├── README.roboflow.txt
├── data.yaml             # Dataset configuration file
├── test/
│   ├── images/           # Test images
│   └── labels/           # YOLO annotations for test images
├── train/
│   ├── images/           # Training images
│   ├── labels/           # YOLO annotations for training images
│   └── labels.cache      # Cache file generated by YOLOv8
└── valid/
    ├── images/           # Validation images
    ├── labels/           # YOLO annotations for validation images
    └── labels.cache      # Cache file generated by YOLOv8
```

Dataset Configuration (css-data/data.yaml)

The YOLOv8 training process uses a configuration file named data.yaml to define the dataset paths, the number of classes, and the class names. This file is a required input for the model.train() function.

Preprocessing

In this project, preprocessing is an essential part of the training pipeline to ensure that all images and their annotations are formatted correctly for the model. The preprocessing steps include:

- Resizing all images to a fixed size of 640x640 pixels (as required by YOLOv8).
- Normalizing pixel values for consistent input to the neural network.
- Reading YOLO-format annotation files for each image and preparing the bounding box data.

These preprocessing steps are handled automatically by the YOLOv8 framework when training begins. There is no need for a separate manual preprocessing script.

Model Training

The machine learning model in this project is trained using YOLOv8 Nano (yolov8n.pt) from the Ultralytics library, which is optimized for fast training and lightweight deployment. The training process is initiated using the script:

preprocess_images.py

How preprocess_images.py Works:

```python
from ultralytics import YOLO

# Load the YOLOv8 Nano model (pre-trained on COCO)
model = YOLO('yolov8n.pt')

# Start training (preprocessing + training)
model.train(
    data='css-data/data.yaml',  # Points to dataset config
    epochs=50,                  # Number of training cycles
    imgsz=640,                  # Input image size (640x640)
    batch=16,                   # Number of images per batch
    name='safety_model32'       # Name of this training run
)
```

Output of the Script

After running the script, YOLOv8 generates:

- The trained model weights (best.pt and last.pt).
- Training metrics like:
  - Precision
  - Recall
  - mAP (mean Average Precision)
- Training charts and visualizations.

After running the training process using YOLOv8, the model outputs are automatically saved in the runs/detect/ directory. This folder contains:

```
runs/
└── detect/
    ├── safety_model322/   # The folder of the current training run (name
    defined in the script)
    └── val/               # Validation results folder (created automatically)
```

Inside the safety_model322/ Folder:

- weights/
  - best.pt – The best model weights based on validation performance.
  - last.pt – The weights from the last epoch of training.
- results.csv – A visual summary of training metrics (loss, precision, recall, mAP over epochs).
- args.yaml – The hyperparameters used for training.
- train_batch*.jpg – Example batch images with detected bounding boxes during training.
- events.out.tfevents.* – TensorBoard logs (optional, if TensorBoard is used).

Model Evaluation

After training is completed, the trained model can be evaluated using YOLOv8's .val() method to check its performance on the validation or test dataset.

Evaluation Script – train_model.py

Despite the name, the train_model.py script is not used for training. Its purpose is to evaluate the trained model (best.pt) on the validation dataset.

Contents of train_model.py:

```
from ultralytics import YOLO

# Load the trained model weights from the previous training run

model = YOLO('runs/detect/safety_model322/weights/best.pt')

# Run evaluation (validation) to measure performance

model.val()
```

Inference / Prediction – Safety Analysis Script (analyze_image.py)

The system includes a Python script named analyze_image.py, which handles the process of making predictions on new images to detect unsafe conditions on construction sites.

Purpose of analyze_image.py:

- **Receives an image path as input (from the PHP system or manually).**
- **Uses the trained model (best.pt) to analyze the image.**
- **Detects safety-related objects (e.g., Hardhat, NO-Hardhat, Mask, NO-Mask, etc.).**
- **Highlights only unsafe detections visually (NO-Hardhat, NO-Mask, NO-Safety Vest).**
- **Counts the number of detected unsafe items.**
- **Outputs:**
  - **The analyzed image with bounding boxes.**
  - **A JSON file containing the class counts.**
  - **Returns the output image path and unsafe count.**

How It Works:

1. Loads the trained YOLOv8 model:

   model = YOLO('runs/detect/safety_model322/weights/best.pt')

2. Receives the image path from command-line argument:

   python3 analyze_image.py path_to_image.jpg

3. Performs detection on the input image.
4. Draws bounding boxes on unsafe detections only.
5. Saves the analyzed image in:

   img/after_analyzing/with_boxes1.jpg (or next available number)

6. Saves detection counts in a JSON file:

   class_counts.json

7. Returns the analyzed image path and unsafe count as JSON output.

HOW TO RUN THE SYSTEM (Installation & Setup)

This section describes all the tools, libraries, and steps required to successfully set up and run the BuildTech system on your local machine.

Required Software & Tools

Tool / Library	Purpose
XAMPP	Provides Apache server and MySQL database
Python 3.x	Runs the Machine Learning scripts
pip	Python package manager (comes with Python)
Composer (PHP)	Manages PHP libraries like mPDF
Git	For version control

Project Files Setup

- Download or clone the project from GitHub.
- Place the project folder inside your XAMPP htdocs directory: /Applications/XAMPP/xamppfiles/htdocs/buildtech/
- or C:\xampp\htdocs\buildtech\ on Windows

Database Setup (MySQL via phpMyAdmin)

1. Start XAMPP: enable Apache and MySQL.
2. Open http://localhost/phpmyadmin.
3. Create a new database named buildtech.
4. Import the database file from:

   /buildtech/buildtech.sql

5. Confirm that tables such as users, projects, and project_images are created successfully.

Python Environment Setup

Check Python Installation:

```
python --version       # macOS/Linux
where python           # Windows CMD
```

Find your Python path and configure it in check_images.php:

```
$pythonPath = "C:\\Users\\yourname\\AppData\\Local\\Programs\
\Python\\Python311\\python.exe";
```

Install Python Dependencies:

From the terminal in the python/ folder:

```
pip install ultralytics opencv-python numpy
```

(If using a requirements.txt:)

```
pip install -r requirements.txt
```

PHP Extensions & Composer Setup

Enable the following PHP extensions in your php.ini (XAMPP > Config):

- mbstring
- gd
- json
- mysqli
- openssl

Check Composer Installation:

```
composer --version
```

If mPDF does not work:

```
composer clear-cache
composer remove mpdf/mpdf
composer require mpdf/mpdf
```

Fix TMP folder issue (if needed):

```
rmdir /s /q tmp
mkdir tmp\mpdf
```

Folder Permissions (for macOS/Linux)

Make sure that the img/ and related folders have the correct permissions to allow saving images:

chmod -R 775 /Applications/XAMPP/xamppfiles/htdocs/buildtech/img/

Running the System Locally

1. **Start Apache and MySQL using XAMPP.**
2. **Open your browser:**

   http://localhost/buildtech/
3. **Log in using your admin account (inserted manually into the database if needed).**

Running the Machine Learning Analysis (Optional for Safety Feature)

To automatically analyze images every hour (or manually):

python call_check_images.py
This script:

- **Checks the database for images with status = 'not_analyzed'.**
- **Calls the YOLO detection model via analyze_image.py.**
- **Saves annotated images and updates the database.**
- **Generates and sends PDF safety reports via email.**

System Features

The BuildTech system is a web-based construction project management platform designed primarily for site managers and administrative staff. The main functionalities of the system are:

Key Features:

- **User Management:**
  - **Secure login system for managers and administrators.**
  - **User access control to ensure that only authorized users can access the system.**
- **Project Management:**
  - **Create, update, and delete construction project records.**
  - **Upload images and documents related to each project.**
  - **Track project details including location, progress, and attached files.**
- **Safety Monitoring (Machine Learning Integration):**
  - **Analyze uploaded construction site images using a YOLOv8-based machine learning model.**
  - **Detect safety-related conditions like:**
    - **NO-Hardhat**
    - **NO-Mask**
    - **NO-Safety Vest**
  - **Highlight unsafe conditions visually on the images (bounding boxes).**
- **Automated Report Generation:**
  - **Generate detailed PDF safety reports after analyzing the images.**
  - **Reports include project information, detection results, and unsafe item counts.**

- **Email Notification (PHPMailer Integration):**
  - **Send PDF reports via email to designated recipients.**
  - **Automated email alerts for safety violations.**
- **Image Management:**
  - **Upload images directly through the project details page.**
  - **Store original and analyzed images separately.**
  - **Save detection counts and analysis results in the database.**
- **Version Control & System Updates:**
  - **This is Version 2 of BuildTech.**
  - **Version 1 included basic project management and file uploads.**

Version 2 adds:

- **Machine Learning Safety Analysis:**
  Integrated YOLOv8 Nano model for automated detection of safety violations (NO-Hardhat, NO-Mask, NO-Safety Vest) in uploaded site images.

- **Automated PDF Safety Reporting:**
  Automatic generation of PDF reports after image analysis, showing unsafe conditions with Hebrew language support and Right-to-Left (RTL) formatting.

- **Email Integration Using PHPMailer:**
  Sends safety analysis reports directly via email to relevant project managers or recipients.

- **Scheduled Safety Checks:**
  Runs periodic safety image analysis automatically through scheduled tasks (Cron job or Python scheduling).

- **System Interface in Hebrew:**
  The user interface and generated reports are fully designed for Hebrew language users, including RTL support for both text display and report formatting.

- **Login Security via PHP Sessions:**
  Improved system security with session-based login verification, ensuring that only authenticated users can access the system's internal pages.

Conclusion

The BuildTech system provides an integrated solution for managing construction projects while focusing on maintaining safety at the construction site. The system combines traditional project management tools with modern machine learning techniques to detect safety issues automatically through image analysis.

This document serves as a comprehensive developer guide for setting up, running, and understanding the BuildTech system (Version 2). It describes the system architecture, machine learning model integration, and provides detailed installation and configuration steps.

Future Recommendations:
Possible future improvements could include:

- **Adding a user-friendly dashboard for viewing safety reports.**
- **Allowing customers (not only managers) to check the progress of their projects.**
- **Expanding the model to detect more safety conditions or unsafe behaviors.**

References

The following resources and supporting materials were used and created as part of the BuildTech project:

Project Repository:

- **BuildTech System (GitHub Repository):**
  https://github.com/lenane68/buildtech.git
  Contains the full source code of the BuildTech system, including backend (PHP), frontend (HTML, CSS, JavaScript), machine learning scripts (Python), and database structure.

Dataset Source:

- **Construction Site Safety Image Dataset (Kaggle):**
  https://www.kaggle.com/datasets/snehilsanyal/construction-site-safety-image-dataset-roboflow
  This dataset was used to train the machine learning model for safety analysis. It includes labeled images of construction workers and equipment with YOLO-format annotations.

Documentation Files:

Document	Description
Developer Guide – Version 2 (This document)	Describes the current system version, architecture, ML integration, setup instructions, and safety analysis process.
User Guide – Version 2	Provides instructions for system users (managers, admins) on how to use the BuildTech system features.
Developer Guide – Version 1	Documentation for the first version of the system, covering basic project management and file upload functionality without machine learning integration.
User Guide – Version 1	Basic usage instructions for the original system version (without ML features).

All guides are available with the project repository or can be requested from the project developer.