Semester SRS Document

Semester Project

Software Requirement Specification

For

Handwritten Urdu Character Recognition

BSCS

By

S#	Name	Registration #/Roll #/Section	Mobile #	E-Mail
1.	M.Fezan	Fall-23-BSCS-466(L)	+92 3097413565	fezan1029@gmail.com
2.	Umer Malik	Fall-23-BSCS-628(L)	+92 3424178351	malikawan56560@gmail.com
3.	Shoaib Rafiq	Fall-23-BSCS-479(L)	+923294423095	shoaibmayo0479@gmail.com

Supervised by:
Sir Mr.Saadat Amir Khan

_____ **(**Signature**)**

Department of Computer Science
Lahore Garrison University

Table of Contents

1. Introduction

1.1 Purpose

This SRS defines the requirements for **URDU-OCR-CNN**, a system for recognizing handwritten Urdu characters and digits using a Convolutional Neural Network (CNN). The system accepts handwritten image input, applies preprocessing, performs prediction via a deep learning model, and delivers results through a web interface and/or backend API.

Product Identification

- **Name:** URDU-OCR-CNN
- **Repository:** H0NEYP0T-466/URDU-OCR-CNN

Technology Stack (Indicative)

- Python (core model training/inference, ~76.5%),
- TypeScript (frontend/UI logic, ~14.6%),
- CSS (styling, ~8%),
- Other (~0.9%)

Scope of This SRS

- End-to-end OCR pipeline for handwritten Urdu characters/digits
- Image preprocessing (normalization, binarization, resizing, augmentation)
- CNN model architecture, training, validation, and model management
- Backend inference API (Python)
- Frontend UI for uploading/drawing characters and viewing predictions

This document focuses only on the OCR subsystem and demo application. Enterprise MLOps features or non-Urdu OCR are out of scope unless stated otherwise. It primarily covers the Python backend and the web frontend. The SRS reflects the master branch as of **2025-12-07** and may require updates as the repository evolves.

1.2 Document Conventions

This document follows standard typographical conventions to ensure clarity and readability:

- **Main Body Text:** Written in **Times New Roman, size 11**.
- **Main Headings:** Displayed in **Blue, size 16** to identify major sections.
- **Sub-headings:** Displayed in **Bold, size 11** to distinguish subsections.
- **Requirement IDs:** Functional requirements are marked with unique identifiers (e.g., **REQ-1**) for traceability.

- **Priorities:** Priorities assigned to high-level features are inherited by their detailed requirements unless explicitly stated otherwise.
- **Technical Terminology:** Standard capitalization is used for specific technologies (e.g., Python, FastAPI, TensorFlow) and file formats (e.g., PNG, JSON).

1.3 Intended Audience and Reading Suggestions

Intended Audience

This SRS is intended for stakeholders involved in the development, evaluation, use, and maintenance of the **URDU-OCR-CNN** system, including:

- **Project Supervisors, Instructors, and Evaluators:** To understand system objectives, scope, constraints, and to assess technical depth, correctness, and completeness.
- **Backend Developers:** To guide model integration, APIs, preprocessing, inference workflow, data flow, performance, and security.
- **Frontend Developers:** To understand user interaction, UI behavior, API usage, and features such as image upload, drawing canvas, and prediction display.
- **Testers and QA Personnel:** To define expected behavior, non-functional requirements, validation criteria, and testable conditions for the web application and APIs.
- **Future Maintainers or Contributors:** To serve as a reference for extending features, optimizing performance, improving models, or refactoring without altering system intent.
- **End Users / Indirect Audience:** To gain a high-level understanding of system capabilities, supported inputs, and limitations.

Document Organization and Reading Suggestions

The SRS supports both high-level understanding and technical reference:

- Section 1: Introduction - Purpose, scope, definitions, audience, and document organization. Recommended for all readers.
- Section 2: Overall Description - System context, user characteristics, operating environment, constraints and assumptions. Recommended for supervisors, developers and maintainers.
- Section 3: Specific Requirements-functional requirements, which include image upload, canvas input, and prediction APIs; non-functional requirements, including performance, usability, and security; and requirements related to data and interface. Recommended for developers and testers.

Readers new to the system should begin with Sections 1 and 2, while technical readers may focus on Section 3. Evaluators should emphasize scope, constraints, and clarity.

1.4 Product Scope

URDU-OCR-CNN is a full-stack application for recognizing handwritten Urdu characters and digits from uploaded images or canvas drawings. It uses Convolutional Neural Networks (CNNs) to analyze handwritten input and outputs predictions with confidence scores. The system is designed to be

accessible, reliable, and extensible, enabling offline handwritten Urdu recognition and bridging the gap between handwritten and digital Urdu text, considering challenges such as cursive structure, right-to-left writing, and character variability.

Objectives

- Accurately recognize handwritten Urdu alphabets and digits from images and canvas input.
- Provide a responsive web interface with real-time feedback.
- Expose prediction functionality via RESTful APIs for integration.
- Ensure modularity and extensibility for future enhancements.
- Support experimentation, evaluation, and learning in handwriting recognition.

Benefits

- Enables digitization of handwritten Urdu, reducing manual transcription.
- Supports educational, academic, and research use cases.
- Serves as a foundation for future word- or document-level OCR.
- Encourages open-source collaboration through transparency and modular design.

Academic and Organizational Alignment

The project applies software engineering best practices to a real-world problem by integrating machine learning into a complete, scalable, and well-documented system. It also contributes to digital inclusion by supporting an underrepresented language like Urdu.

2. Overall Description

2.1 Product Perspective

URDU-OCR-CNN is a new, self-contained full-stack web application for handwritten Urdu character and digit recognition. It is not a replacement or extension of any existing system, but an independent product that integrates modern web technologies with a trained convolutional neural network for prediction.

The system follows a **client–server architecture** and consists of three main components:

- **Frontend Client:** A browser-based interface that allows users to upload images or draw characters on a canvas, handles user interaction and validation, and displays prediction results.
- **Backend API Server:** Processes requests from the frontend, performs image preprocessing, runs CNN-based inference, and returns predictions with confidence scores.
- **Model and Data Layer:** Contains trained CNN models, preprocessing pipelines, label mappings, and datasets, and operates independently of the user interface, allowing updates or retraining without frontend changes.

System Interaction Flow:
Users interact with the frontend, which sends image data to the backend via HTTP. The backend performs inference using the CNN model and returns the results, which are then displayed to the user. Prediction functionality is also exposed through documented API endpoints for potential external integration.

While the current scope focuses on single-character and digit recognition, the architecture is modular and extensible, supporting future enhancements such as word-level recognition or integration into larger document-processing systems. Although academic in nature, the system design follows real-world best practices, including modularity, clear separation of concerns, and deployment readiness.

2.2 Product Functions

URDU-OCR-CNN provides the following high-level functions:

- **Handwritten Input Acquisition:** Users can submit Urdu characters/digits via image upload or an interactive browser canvas.
- **Image Preprocessing:** Input images are automatically resized, normalized, and validated for model compatibility.
- **Character Recognition:** Preprocessed inputs are analyzed using CNN models to identify Urdu alphabets and digits.
- **Prediction Generation:** Outputs include the most likely character and confidence scores.
- **Top-K Prediction Display:** Multiple likely predictions are presented to improve usability and interpretability.
- **Web-Based User Interface:** Responsive interface for input and result visualization.
- **RESTful API Services:** Exposes prediction functionality via documented endpoints for programmatic integration.
- **System Monitoring and Health Checks:** Provides basic service availability and status endpoints.
- **Model Management:** Allows loading, updating, or replacing models without frontend changes.
- **Testing and Validation Support:** Facilitates automated verification of system behavior.

These functions enable conversion of handwritten Urdu inputs into structured digital outputs while maintaining **usability, modularity, and extensibility**. Detailed functional requirements are described in **Section 3**.

2.3 User Classes and Characteristics

URDU-OCR-CNN serves multiple user classes with varying technical expertise and system needs:

Primary Users

- **General Users (End Users):**

    - **Traits:** Limited technical background, familiar with basic web apps, interested in recognizing handwritten Urdu.
    - **Usage:** Upload images or draw on canvas, view predictions with confidence scores.
    - **Importance:** Highest priority; usability and accuracy directly affect experience.
- **Frontend and Backend Developers:**
    - **Traits:** Strong technical skills in web development, APIs, and system architecture.
    - **Usage:** Modify frontend/backend, integrate or refine models and features.
    - **Importance:** High priority; system designed for modularity and extensibility.
- **Testers / QA Users:**
    - **Traits:** Technical or semi-technical, familiar with testing workflows.
    - **Usage:** Test API endpoints, UI, and validate functional/non-functional requirements.
    - **Importance:** Medium; support reliability and correctness.

Secondary Users

- **Researchers / Students:**
    - **Traits:** Academic or research-oriented, interested in handwriting recognition.
    - **Usage:** Analyze model behavior, system architecture, and experiment.
    - **Importance:** Lower; benefit from system transparency but not core operational users.
- **System Maintainers / Administrators:**
    - **Traits:** Technical background, responsible for deployment, monitoring, and updates.
    - **Usage:** Deploy system, monitor health endpoints and logs.
    - **Importance:** Lower; focused on maintenance.

User Class Prioritization:

- **Highest:** End Users
- **High:** Developers
- **Medium:** Testers
- **Low:** Researchers & Maintainers

Usability requirements target end users, while interface, performance, and extensibility requirements are more relevant for developers and testers.

2.4 Operating Environment

URDU-OCR-CNN operates in a distributed web-based environment with a frontend hosted publicly and a self-hosted backend. Standard web technologies are used, requiring no special client-side installations.

Client-Side (Frontend)

- **Deployment:** Publicly accessible web platform.
- **Hardware:** Computer, laptop, or handheld with internet access.
- **Operating Systems:** Any OS supporting a modern web browser (Windows, macOS, Linux, Android, iOS).
- **Web Browsers:** Latest stable versions of Chrome, Firefox, Edge, or Safari.

Server-Side (Backend – Self-Hosted)

- **Deployment:** Locally or privately managed server.
- **Hardware:** CPU sufficient for image processing and inference; GPU optional for model training.
- **Operating Systems:** Preferably Windows; tested on Windows and macOS.
- **Runtime:** Python 3.10+
- **Backend Services:** RESTful API handling image preprocessing and prediction, local model storage, and inference execution.

Coexistence Requirements

- Must coexist with other applications without resource conflicts.
- Frontend-backend communication requires available network ports.

2.5 Design and Implementation Constraints

Hardware Constraints:

- Backend requires sufficient CPU for real-time image processing and inference.
- GPU is optional for training; not required for runtime predictions.
- Frontend operates on standard client devices without custom hardware.

Software and Platform Constraints:

- Backend runs on Python 3.10+.
- Frontend runs on modern browsers (Chrome, Firefox, Edge, Safari).
- Backend is self-hosted; frontend is web-deployed.

Interface Constraints:

- Backend exposes a RESTful API for predictions and system status.

- Frontend communicates via HTTP/HTTPS only.
- Inference relies solely on local models; no third-party APIs.

Design and Programming Standards:

- Backend follows PEP 8 and modular design.
- Frontend adheres to React and TypeScript best practices.
- All modules maintain requirement IDs, priorities, and traceability per SRS.

Data and Security Constraints:

- Input images must be PNG or JPG.
- System handles Urdu's right-to-left text correctly.
- No personal data is stored; user privacy is maintained.

Operational Constraints:

- Predictions must be returned in near real-time.
- Model retraining is offline and optional; inference must not block frontend responsiveness.
- Deployment aligns with the described environment, requiring no additional infrastructure.

2.6 User Documentation

The URDU-OCR-CNN system will be provided with comprehensive user documentation intended to help developers, testers, and end-users in the installation, using, and maintenance of the software. All the documents are based on the standard community and open-source conventions.

Included Documentation Components

- README.md – Overview, features, prerequisites, installation and usage instructions.
- SETUP.md - Details setup guide for backend and frontend environments.
- API.md - Full API reference including backend endpoints, request/response formats, and example usages.
- CONTRIBUTING.md – Contribution guidelines including code style, pull requests, and testing procedures.
- ISSUE TEMPLATES – Standartized bug report and feature request templates for GitHub repository.
- DISCUSSIONS / Community Guidelines–Directions for community support, feedback, and collaboration.

Delivery Formats and Standards

- Markdown format (.md) for GitHub-hosted documentation.

- All documents are web-accessible via the repository, supporting search, navigation and versioning.
- Documentation follows the standards of open source communities for readability, maintainability, and completeness.

2.7 Assumptions and Dependencies

Assumptions:

- Users provide reasonably clear images (not blurred or corrupted) for accurate recognition.
- Backend has sufficient CPU for real-time inference; GPU is optional for training.
- Developers/maintainers use Python, FastAPI, and relevant ML libraries.
- Training datasets represent real-world Urdu handwriting variations.
- Internet is needed only for initial setup and dependency installation; offline inference does not require connectivity.

Dependencies:

- **Software Libraries:** TensorFlow, NumPy, OpenCV, scikit-learn, and other packages in requirements.txt.
- **Datasets:** UNHD, UCOM, Kaggle Urdu handwriting datasets, or other publicly available sources.
- **Operating Environment:** Python 3.10+ on Windows, Linux, or macOS.
- **System Conventions:** Git/GitHub for repository management, version control, and documentation.

Notes:

- Violation of assumptions (e.g., low-quality images, missing datasets) may degrade performance.
- Dependencies are version-sensitive; updates to Python packages or datasets may break functionality.

3. External Interface Requirements

3.1 User Interfaces

URDU-OCR-CNN provides a web-based interface supporting image uploads and canvas input.

Frontend Interface

- **Web Application:** Accessible via standard browsers on desktop or mobile.
- **Main Components:**

- **Home Page:** Introduction, upload options, navigation.
- **Image Upload Module:** Drag-and-drop or click-to-upload handwritten characters.
- **Drawing Canvas:** Draw Urdu characters with clear and undo controls.
- **Prediction Panel:** Displays top-5 predicted characters with confidence scores.
- **Navigation & Footer:** Menu (About, How It Works, Documentation) and contact/version info.

GUI Standards

- **Layout:** Responsive, right-to-left alignment for Urdu, consistent CSS styling.
- **Colors & Fonts:** High-contrast, readable fonts; primary color #111 (black), secondary amber for Urdu characters.
- **Buttons & Controls:** Standardized Upload, Submit, Clear with hover effects.
- **Error/Feedback Messages:** Shown near input areas for invalid file, empty canvas, or prediction failure.

User Interface Scope

- **General Users:** Upload/draw characters and view predictions.
- **Administrators/Developers:** Access logs or API endpoints via documentation (not GUI).

Detailed wireframes and UI design elements are documented separately in docs/UI_SPEC.md.

3.2 Hardware Interfaces

URDU-OCR-CNN interacts with hardware for computation, storage, and user input.

Supported Devices

- **Server/Workstation:** Hosts backend, inference, and optional training.
  - Minimum: 8 GB RAM, Quad-core CPU, 50 GB disk.
  - Recommended: 16+ GB RAM, GPU (CUDA-compatible) for faster training.
- **Client Devices:** Standard desktops, laptops, or mobiles with modern browsers.
- **Input Devices:** Mouse, keyboard, touch screen for canvas input.

Hardware Interaction

- **Data Transfer:** Images and canvas drawings uploaded via HTTP/HTTPS POST; canvas serialized to PNG.
- **Computation:** CNN inference performed on server; optional GPU acceleration via CUDA/OpenCL.
- **Storage:** Local or network storage for images, processed data, models, and logs.

Communication Protocols

- Client ↔ Server: HTTP/HTTPS over TCP/IP.
- Backend internal: File system and memory-based; no direct hardware bus access.
- Optional GPU: CUDA/OpenCL APIs.

Constraints

- Prediction speed depends on CPU/GPU and available memory.
- Storage must accommodate model checkpoints, logs, and datasets.
- System must handle hardware limits gracefully with appropriate error messages.

3.3 Software Interfaces

URDU-OCR-CNN interacts with multiple software components, libraries, and services for data processing, model inference, and user interaction.

Operating Systems

- Windows 10+, Linux Ubuntu 20.04+, macOS 12+
- Provides standard filesystem, process, and network services.

Backend Framework & Libraries

- **Python 3.10+** – main language
- **FastAPI** – HTTP/HTTPS handling, routing, REST API v0.100+
- **TensorFlow 2.14+** – CNN model definition, training, inference
- **NumPy 1.26+** – numerical computations
- **OpenCV 5.3+** – image processing and transformations
- **scikit-learn 1.3+** – evaluation metrics and data splitting
- **Pydantic 2.5+** – request/response data validation

Data Storage

- File-based storage for dataset images, processed images, trained model files (.h5), label JSONs, and logs.
- Shared data includes uploaded images, preprocessed arrays, and prediction results (top-N characters with confidence scores).

Component Communication

- **Client ↔ Backend:** JSON over HTTP/HTTPS; images sent as multipart/form-data or base64 PNG.
- **Backend Modules:**

    - image_service – preprocess and normalize images
    - model_service – CNN inference
    - helpers – shared utilities
    - Logging/config: .env and config.py, logs stored locally

External Dependencies

- Datasets: UNHD, UCOM, Kaggle Urdu handwritten datasets
- Library versions must be compatible; updates may require retraining or code adjustments

Services

- REST API endpoints:
    - /api/v1/predict – accepts images, returns predictions
    - /api/v1/predict/canvas – accepts canvas input, returns predictions
    - /api/v1/classes – returns supported Urdu characters and digits
- Detailed API specs in docs/API.md

3.4 Communications Interfaces

URDU-OCR-CNN communicates primarily between frontend and backend via standard web protocols.

- **Protocol:** HTTP/HTTPS for all API requests.
- **Endpoints:** RESTful APIs: /api/v1/predict, /api/v1/predict/canvas, /api/v1/classes.
- **Message Format:**
    - Image uploads: multipart/form-data
    - Metadata and predictions: JSON, including top-N predictions and confidence scores.
- **Security:**
    - Optional HTTPS for secure data transfer
    - No authentication in current version; future versions may use API keys.
- **Data Transfer:** Images <1 MB; standard broadband or LAN sufficient.
- **Synchronization:** Stateless requests; concurrent requests handled independently by FastAPI.

4. System Features

4.1 System Feature 1: Handwritten Urdu Character Recognition

Description & Priority:

- Allows users to upload images or draw characters on a canvas to obtain predictions of Urdu letters or digits.
- **Priority:** High

Stimulus/Response Sequences:

- **Upload image:** System preprocesses → feeds to CNN → returns top-N predictions with confidence.
- **Draw on canvas:** System captures → preprocesses → predicts → displays results.

Functional Requirements:

- **REQ-1:** Accept PNG, JPEG, BMP images of handwritten characters.
- **REQ-2:** Preprocess images to 64×64 grayscale arrays.
- **REQ-3:** Return top-5 predictions with confidence scores.
- **REQ-4:** Display errors for unsupported formats or corrupted files.
- **REQ-5:** Allow multiple consecutive predictions without restarting.

4.2 System Feature 2: Canvas-based Drawing Recognition

Description & Priority:

- In-browser drawing interface for instant recognition of handwritten Urdu characters.
- **Priority:** Medium

Stimulus/Response Sequences:

- **Draw character:** System preprocesses → CNN predicts → displays prediction.

Functional Requirements:

- **REQ-6:** Capture canvas input as base64-encoded PNG.
- **REQ-7:** Preprocess and normalize canvas images before inference.
- **REQ-8:** Provide immediate visual feedback of predictions.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Predictions returned within 100–300 ms per image.
- Preprocessing and inference optimized for minimal memory usage (<2 GB).

5.2 Safety Requirements

- No physical hazards exist because it is a purely software-based system.

- It prevents Input Validation from processing malformed or malicious image files.

5.3 Security Requirements

- The recommended HTTPS will be used for secure data transfer.
- Optional API authentication for future enterprise deployment.
- Local storage of uploaded images is temporary; files are deleted after inference.

5.4 Software Quality Attributes

- Availability: 99% uptime for hosted backend.
- Maintainability: Modular backend design using Python modules.
- Portability: Runs on Windows, Linux, macOS.
- Usability: Clear API responses; web interface is intuitive.
- Robustness: Graceful handling of invalid inputs and exceptions.

5.5 Business Rules

- Only supported Urdu characters and digits are processed.
- Predictions returned are for recognition purposes; no validation of user input required.
- Users may submit unlimited requests but must comply with rate-limiting if applied in deployment.

References

- URDU-OCR-CNN Github Repository: https://github.com/H0NEYP0T-466/URDU-OCR-CNN
- Dataset: https: https://www.kaggle.com/datasets/surindersinghkhurana/handwritten-urdu-characters-dataset
- TensorFlow Documentation: https://www.tensorflow.org/api_docs
- FastAPI Documentation: https://fastapi.tiangolo.com/
- OpenCV Documentation: https://opencv.org/
- Other Dataset References: https://www.youtube.com/@Alifbaypay1