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 describes the requirements for the "URDU-OCR-CNN" system, a solution for the recognition of handwritten Urdu characters using a Convolutional Neural Network (CNN). The product shall take in an image of any handwritten Urdu character or digit, perform pre-processing, use a deep learning model to make a prediction, and output the predictions both through a front-end user interface and/or back-end API.

Product identification:

Product Name : URDU-OCR-CNN

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

Technology composition (indicative):

Python (core model training/inference, ~76.5%),

TypeScript (frontend/UI logic, ~14.6%),

CSS (styling, ~8%),

Other (~0.9%)

Scope covered by this SRS:

End-to-end OCR subsystem for handwritten Urdu characters/digits:

Data handling/preprocessing pipeline: normalizing images, binarizing, resizing, and augmenting.

Model Architecture: CNN-based; Model Training; Model Validation; Model Artifact Management.

- Inference services: Python backend API for prediction.

Upload or draw characters and view predictions - UI for Frontend in TypeScript/CSS.

It focuses on the OCR subsystem and integrated demo application alone; it does not cover enterprise-grade MLOps, or multilingual OCR beyond Urdu, unless mentioned explicitly in later sections.

- If the repository contains several components, then this document mainly describes:

- The Python backend (model and inference service) under backend/- The web frontend under src/, public/, index.html and Vite configuration.This SRS reflects the current snapshot of the master branch as of 2025-12-07. Further changes to the repository may necessitate revising and versioning the SRS.

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

Intended Audience

- This SRS targets the following readers in relation to the development, evaluation, use, and maintenance of the URDU-OCR-CNN system:
- Project Supervisors, Instructors, and Evaluators: Explain system objectives, scope, constraints, and the requirements for reviewing technical depth, correctness, and completeness.
- Backend Developers: Guide server-side logic, model integration, APIs, data processing, performance, security, and data flow for inference, preprocessing, and endpoints.
- Frontend Developers: Explain how users will interact with an interface, how the interface will behave, how an API will be used, and what is expected as an outcome for features such as image upload, drawing canvas, and predictions.
- Testers and QA Personnel: Expected behavior definition, non-functional requirements, validation criteria, and testable conditions for web app and API services.
- Future Maintainers or Contributors: Use this as a reference while implementing features, optimizing performance, enhancing models or refactoring without changing the system's intent.
- Indirect Audience/End-Users: Convey high-level understanding of system capabilities, supported methods of input, 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.

Appendices and Supporting Sections –Glossary, traceability, and supplementary technical details. Recommended for evaluators and future contributors. Readers who are unfamiliar with the system should read Sections 1 and 2 for background and then refer to Section 3 as needed.

Technical readers may want to focus on requirements; evaluators on scope, constraints, and clarity.

1.4 Product Scope

The URDU-OCR-CNN is a full-stack software application capable of recognizing handwritten Urdu characters/digits from given input images/drawings provided by the user. It uses CNNs to analyze the handwritten input and produces accurate character predictions with confidence scores.

The foreseen role of this software is to provide an accessible, reliable, and extensible platform for offline handwritten Urdu character recognition. The system aims to bridge the gap between traditional handwritten text and digital text representation for the Urdu language, which presents unique challenges due to its cursive, right-to-left script, and variability in characters.

Objectives and Goals

Key objectives of the URDU-OCR-CNN system are:

→Accurately recognizing handwritten Urdu alphabets and digits from scanned images and freehand canvas input.

→Providing a responsive web-based interface that allows for multiple methods of inputting information and real-time feedback.

→To expose the prediction functionality through well-defined RESTful APIs for integration with other applications.

→Modularity and extensibility to allow enhancement of models, datasets, and interfaces in the future.

→To support experimentation, evaluation, and learning within the area of handwriting recognition and intelligent systems.

Benefits

- This allows the digitization of written Urdu content, removing the need for manual transcription.
- Facilitates various educational, academic, and research use cases related to Urdu text processing.
- Provides the base that can be extended to word level or document-level recognition in the future.

- It encourages opensource collaboration and reproducibility by being transparent, well-documented, and highly modular.

Alignment with Organizational and Academic Goals

The project aligns with academic goals of applying software engineering principles in solving real-world problems, integrating a machine learning model into a scalable software system, and delivering a complete, documented solution following all the best practices of standard SRS. It also furthers larger technological goals of promoting digital inclusivity for underrepresented languages such as Urdu.

2. Overall Description

2.1 Product Perspective

The URDU-OCR-CNN system is an entirely new, self-contained software product being developed as a full-stack web application for handwritten Urdu character and digit recognition. It does not replace any existing production system; neither is it a straight-forward extension of any previous product family. Instead, it will be an autonomous system that integrates modern web technologies with a trained convolutional neural network for prediction services.

The product follows a client–server architecture and is made up of three major logical components:

Frontend Client

A browser-based web interface is provided through which the user can upload the handwritten images or draw the characters on an interactive canvas. The frontend handles user interactions, input validations, and shows the results of the prediction.

Backend API Server

This web service accepts incoming requests from the frontend, does some preprocessing of the images, invokes trained CNN models for inference, and returns prediction results along with confidence scores.

Model and Data Layer

This layer contains the trained CNN models, preprocessing pipelines, class label mappings, and datasets used for training and evaluation; it operates independently of the user interface and can be updated or retrained without modifications to the front end.

The relationship among these elements is conceptualized as follows:

- The user interacts with the frontend interface.
- The image data is sent from the frontend to the backend API via HTTP requests.
- The backend processes the input, executes the inference using the CNN model, and returns the prediction results.

The frontend shows the results to the user in a readable format. It also exposes its functionality via documented API endpoints, in case there will be a future need to integrate the system with external applications or services. Although the current project scope focuses on single-character and digit recognition, the architecture is modular and extendable. This allows future enhancements, such as word-level recognition, additional scripts, or integration into larger document-processing systems without fundamental architectural changes. URDU-OCR-CNN is an independent academic and research-oriented system, yet design choices align well with the best practices for its real-world deployment, such as modular components, containerization support, and clear separation of concerns.

2.2 Product Functions

The URDU-OCR-CNN system provides the following major functions at a high level:

Handwritten Input Acquisition

Allows users to provide handwritten Urdu characters/digits through uploading image files or by drawing directly onto an interactive browser-based canvas.

Image Preprocessing

Automatically processes input images to ensure compatibility with the recognition model by resizing, normalization, and format validation.

Handwritten Character Recognition

Analyzes the preprocessed input to recognize Urdu alphabets and digits using trained convolutional neural network models.

Generation of Predicted Results

Generates prediction outputs including the most likely recognized character and associated confidence scores.

Top-K Prediction Display

Presents the user with a variety of likely predictions to enhance interpretability and usability.

Web-Based User Interface The interaction interface provides responsiveness and a user-friendly gateway to enter inputs and visualize results. RESTful API Services Exposes prediction functionality via documented API endpoints that are accessible programmatically and able to be integrated with external systems. System Monitoring and Health Checks Provides basic system health endpoints for service availability and status verification. Model Management Support Allows loading, updating, and replacing the different trained models without changing anything in the frontend interface. Testing and Validation Support Supports system behavior verification using automated testing mechanisms. These features collectively enable the system to convert handwritten Urdu character inputs into structured digital outputs while maintaining usability,

modularity, and extensibility. The detailed functional requirements for each capability are outlined in Section 3 of this document.

2.3 User Classes and Characteristics

The URDU-OCR-CNN system is intended for multiple user classes with varying levels of technical expertise, usage frequency, and access to system functionality. These user classes are described below, along with their key characteristics.

Primary User Classes

General Users (End Users)

These are major users of the system who interact directly with the web interface.

Traits:

- Limited or no technical background
- Familiar with basic web applications
- Interested in recognizing handwritten Urdu characters or digits.

Usage Patterns:

- Upload handwritten images or draw characters on the canvas.
- View Predicted Characters with Confidence Scores

Importance:

This is the most important user class, as system usability and correctness directly impact their experience.

Frontend and Backend Developers

The developers are responsible for building, maintaining, and extending the system.

Characteristics:

- Strong technical background
- Familiarity with web development, APIs, and software architecture

Usage Patterns:

- Modify either frontend interfaces or backend logic
- Integrate new models or refine the existing functionalities

Importance

High importance, as the system is designed to be modular and extensible for future development.

Testers / Quality Assurance Users

These users ensure that the system acts as expected.

Characteristics

- Technical or semi-technical background
- Familiarity with testing tools and validation workflows

Usage Patterns:

- Test API endpoints and UI behavior
- Confirm functional and non-functional requirements

Importance:

Low to medium importance: supporting system reliability and correctness.

Secondary User Classes

Researchers/Students

This class includes academic users who may study or extend the system for learning or research purposes.

Traits:

- Academic or research-oriented background
- Interest in handwritten recognition systems or system design

Usage Patterns:

- Analyze model behavior and system architecture
- Use the system as a baseline for experimentation

Importance:

Secondary importance, as they are not core operational users but benefit from system transparency.

System Maintainers / Administrators

These users manage the deployment, configuration, and system availability.

Characteristics:

- Technical background
- Responsible for system setup, monitoring, and updates

Usage Patterns:

- Deploy the system
- Health endpoint and log monitoring

Importance:

- Of secondary importance, only offering maintenance.
- User Class Prioritization Summary

Highest Priority: End Users

High Priority: Developers

Medium Priority: Testers Lower Priority: Researchers and Maintainers Certain requirements, particularly usability-related requirements, primarily target end users, while interface, performance, and extensibility requirements are more relevant to developers and testers.

2.4 Operating Environment

The proposed system URDU-OCR-CNN works in a distributed web-based environment where the frontend application is deployed and backend service self-hosted. The system has been designed to use standard web technologies for access without special installations at the client side.

Client-Side Environment (Frontend)

Deployment:

The frontend is hosted on a publicly accessible web platform.

Hardware:

A computer, laptop, or handheld that has access to the internet.

Operating System:

Any operating system that can run a recent web browser including:

Windows || macOS || Linux || Android || iOS

Web Browser:

Latest stable versions of:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

Server-Side Environment (Backend - Self-Hosted)

Deployment:

The service back-end is self-hosted on a locally or privately managed server.

Hardware:

- CPU-based system sufficient for processing all image processing and inference tasks.
- Optional GPU support for model training; runtime (inference) does not require it.

Operating System:

- Systems based on Windows are preferred.
- The project is developed and tested under both Windows and macOS.

Runtime Environment:

Python 3.10 or later

Backend Services:

- RESTful API service responsible for image processing and prediction handling.
- Local model storage and runtime inference execution.
- Coexistence Requirements
- The service has to coexist with other applications running on the same host system without resource conflicts.
- The API communication between the frontend and backend requires available network ports.

2.5 Design and Implementation Constraints

The design and implementation of the URDU-OCR-CNN system is driven by several constraints:

Hardware Constraints

- The backend server needs sufficient CPU resources to handle real-time image processing and model inference.
- Model training can optionally use GPU acceleration (though it is not required for runtime predictions).
- Frontend shall operate on standard client devices without custom hardware.

Software and platform constraints

- The backend should run on Python 3.10+.
- The Frontend should run on modern web browsers: Chrome, Firefox, Edge, Safari.
- The system has to operate within a self-hosted environment for the backend, and a deployed web environment for the frontend.

Interface Constraints

- The backend provides a RESTful API with image prediction and system status endpoints.

- Frontend communicates with backend only via HTTP/HTTPS requests.
- Inference shall not be based on any external third-party APIs.

Design and Programming Standards

- The backend code adheres to PEP 8 standards and a modular structure to make it maintainable.
- Frontend code follows the best practices of React and TypeScript conventions.
- All modules should maintain the requirement IDs, priorities, and traceability for alignment with SRS.

Data and Security Constraints

- Input images should be in standard formats such as PNG and JPG.
- The system has to handle the right-to-left text of Urdu correctly.
- User data and predictions are treated in conformance with local norms of privacy; no personal information is ever stored.

Operational Constraints

- The system needs to process images and return predictions in near real-time.
- Model retraining is offline and optional. Inference mustn't block frontend responsiveness.
- Deployment should be consistent with the existing software stack without requiring additional infrastructure beyond what was described in the environment.

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

The development and operation of the URDU-OCR-CNN system rely on several assumptions and dependencies, which could affect its operation if changed.

Assumptions

- Users will provide input images of reasonable quality, such as not blurred or corrupted, for accurate recognition.
- Backend server has sufficient CPU resources for real-time inference and optional GPU for model training.
- Developers and maintainers work with Python, FastAPI, and relevant ML libraries.
- These handwritten datasets of Urdu, which are used for training, represent real-world handwriting variations.
- Internet connectivity may be required for initial setup and installing dependencies but is not required for offline inference.

Dependencies

Software Libraries: TensorFlow, NumPy, OpenCV, scikit-learn, and other Python packages listed in requirements.txt.
Datasets: UNHD, UCOM, Kaggle Urdu handwritten characters dataset, or other publicly available Urdu handwriting datasets.
Operating Environment: Python 3.10+ on supported operating systems (Windows/Linux/macOS).

System Conventions: The Git and GitHub infrastructures are used for repository management, documentation, and code versioning.
Notes

- Any failure in the assumptions, such as low-quality images or unavailable datasets, may degrade the performance of the system.
- Dependencies are often version-sensitive, and Python package updates or dataset changes can sometimes break code.

3. External Interface Requirements

3.1 User Interfaces

The URDU-OCR-CNN system offers a web-based interface to interact with the software, where image uploads and direct canvas input are supported. The logical characteristics of the user interface are as follows:

1. Frontend Interface

Web Application: Can be used through standard web browsers with desktops or mobile devices.

Main Components:

- Home Page: Introduction, upload options, and navigation to other sections.
- Image Upload Module: A drag-and-drop or click-to-upload functionality for handwritten characters.
- Drawing Canvas: Users can draw Urdu characters directly; basic editing controls include clear and undo.
- Prediction Results Panel - displays top-5 predicted characters with confidence scores.
- Navigation & Footer: Regular navigation menu with About, How It Works, and Documentation; footer with contact info and version.

2. GUI Standards

- Layout: Uniform and responsive design with CSS; supports right-to-left alignment for the Urdu script.
- Colors & Fonts: Clear contrast for readability, fonts consistent and primary color is a shade of black(#111) and secondary color is amber or similar in case of Urdu characters.
- Buttons & Controllers: Standardized across Upload, Submission, Clear; hover effects show actionability.
- Error/Feedback Messages-on the homepage, near the input area; examples include invalid file type, empty canvas, or failure in prediction.

3. User Interface Scope

Interfaces are provided for:

- General Users: Upload or draw characters and view predictions.
- Administrators/Developers: access logs or API endpoints (via documentation), not part of the GUI.

Note: The detailed layout diagrams, wireframes, and interactive design elements are documented separately in the User Interface Specification (docs/UI_SPEC.md).

3.2 Hardware Interfaces

The URDU-OCR-CNN system interacts with hardware mainly for the purpose of computation, storage, and user input. Following are some of the logical and physical characteristics of the hardware interfaces:

1. Supported Device Types

Server/Workstation: Hosts the backend, model inference, and optionally training. Should have:

- Minimum: 8 GB RAM, Quad-core CPU, 50 GB disk space.
- Recommended: 16+ GB RAM, GPU support (NVIDIA CUDA-compatible) for faster model training.
- Client Devices: Users access the system using standard desktops, laptops, or mobile devices capable of running modern web browsers.
- Input Devices: Mouse, keyboard, and touch screen for drawing input on canvas.

2. Hardware Interaction

Data Transfer:

- Images from client devices are uploaded to the backend via HTTP/HTTPS POST requests.
- The resulting canvas drawings are serialized into image format, for example, PNG.

Computation:

- Backend performs inference of CNN models on server hardware.
- Optional GPU acceleration via CUDA/OpenCL if available.

Storage:

The images uploaded, data processed, models trained, and logs are kept on local or networked storage.

3. Communication Protocols

- Client ↔ Server: HTTP/HTTPS over TCP/IP.
- Internal Backend Processing: With file system and memory-based communication, no direct interaction with the hardware bus is required.
- Optional GPU acceleration: CUDA/OpenCL APIs for interfacing with compatible graphics hardware.

4. Constraints

- Real-time prediction performance depends on CPU/GPU capabilities and available memory.
- The storage space should be enough to accommodate model checkpoints, logs, and dataset storage.

- System should handle hardware resource limits (insufficient memory, low disk space, etc.) gracefully with appropriate error messages.

3.3 Software Interfaces

The URDU-OCR-CNN system interfaces with numerous software components, libraries, and tools to conduct the tasks of data processing, model inference, and user interaction.

1. Operating System

- Supported operating systems include Windows 10+, Linux-Ubuntu 20.04+, and macOS 12+.
- Provides standard filesystem, process management, and network services to execute applications.

2. Back-end Framework and Libraries

- Python 3.10+: Main programming language.
- FastAPI: handles HTTP/HTTPS requests, routing, REST API endpoints v0.100+
- TensorFlow (v2.14+): CNN model definition, training, and inference.
- NumPy (v1.26+): Miscellaneous numerical computations and array manipulations.
- OpenCV: Image processing, resizing; transformations (v5.3+).
- scikit-learn (v1.3+): Evaluation metrics and data splitting.
- Pydantic, v2.5+: Validation of data in request and response models.

3. Databases / File Storage

File-based storage: Dataset images, processed images, trained model files (.h5), label JSONs, and logs.

Data Shared Across Components:

- User-uploaded images or canvas drawings.
- Processed Data: Preprocessed image arrays that are fed to CNN for inference.
- Prediction results: returned top-N predicted characters along with confidence scores via API or frontend.

4. Component-Component Communication

Client ↔ Backend: JSON-formatted responses over HTTP/HTTPS; image data sent as multipart/form-data or base64-encoded PNG.

Backend Modules:

- image_service → preprocess and normalize images for model.
- model_service → perform CNN inference.
- helpers → shared utility functions.

- Logging and Configuration: Configurations read from .env and config.py, logs stored locally using standard Python logging.

5. External Dependencies

- Datasets: UNHD, UCOM, Kaggle Urdu handwritten datasets. Used for training and evaluation.
- Version Sensitivity: The version of backend libraries and TensorFlow should be compatible; otherwise, the models may not work. Updates may require retraining or code adjustments.

6. Services Needed

- REST API services for:
- /api/v1/predict → Accept images, return predictions.
- /api/v1/predict/canvas → Accept canvas input, return predictions.
- /api/v1/classes → Return supported Urdu characters and digits.
- Note: Detailed API protocols and request/response schemas are documented in docs/API.md.

3.4 Communications Interfaces

The URDU-OCR-CNN system requires communication primarily between the client (frontend) and backend (API) over standard web protocols:

- Protocol: HTTP/HTTPS for all API requests.
- Endpoints: RESTful APIs (/api/v1/predict, /api/v1/predict/canvas, /api/v1/classes).

Formatting Message:

- Request Body: multipart/form-data for image uploads and JSON for metadata.
- JSON format, with the outcomes of prediction, top-N predictions, and confidence scores.

Security:

- Optional HTTPS encryption for secure data transfer.
- No user authentication required in the current version; future releases may include API keys.
- Data Transfer: Typically small (images <1 MB), requiring standard broadband or LAN connectivity.
- Synchronization: Requests are stateless; concurrent requests handled independently by FastAPI.

4. System Features

4.1 System Feature 1(Handwritten Urdu Character Recognition)

4.1.1 Description and Priority

Description: Allows users to upload images or draw characters on a canvas and obtain predictions of Urdu letters or digits.

High Priority

4.1.2 Stimulus/Response Sequences

User Action: Upload image → System Response: Preprocess image → Feed to CNN → Return top-N predictions with confidence.

User Action: Draw character on canvas → System Response: Capture canvas → Preprocess → Predict → Display results.

4.1.3 Funcional Requirements

- REQ-1: The System shall accept PNG, JPEG, or BMP format images of handwritten characters.
- REQ-2: The system shall preprocess images to 64×64 grayscale arrays.
- REQ-3: System shall return top-5 predictions with confidence scores.
- REQ-4: System shall display error messages for unsupported formats or corrupted files.
- REQ-5: System shall allow multiple consecutive predictions without restarting the service.

4.2 System Feature 2(Canvas-based Drawing Recognition)

4.2.1 Description and Priority

Description: Provides an in-browser drawing interface for instant recognition of handwritten Urdu characters.

Priority Medium

4.2.2 Stimulus/Response Sequences

User Action: Draw character → System Response: Preprocess → CNN predicts → Display prediction.

4.2.3 Functional Requirements

- REQ-6: System shall capture the canvas input as base64-encoded PNG.
- REQ-7: System shall preprocess and normalize the canvas image before inference.
- REQ-8: The system should give immediate visual feedback of the prediction.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- System shall return predictions within 100–300 milliseconds per image.
- Preprocessing and inference pipelines 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