**University of Transport Ho Chi Minh City**

**Faculty of Information Technology**

**Software Requirement**

**Specification**

**for**

**To-do list WebApp**

**Version 1.0**

**Le Thi Lan Huong**

**Nguyen Phuong Quynh Nghi**

**Mai Quynh Tram**

**Mai Quynh Tran**

**Date created 04-12-2024**

# **Table of Contents**

[**Table of Contents** 2](#_Toc190561065)

[1. Introduction 4](#_Toc190561066)

[1.1. Document Purpose 4](#_Toc190561067)

[1.2. Intended Use and Target Audience 4](#_Toc190561068)

[1.3. Product Scope 4](#_Toc190561069)

[1.4. Definitions, Acronyms and Abbreviations 4](#_Toc190561070)

[1.5. References 5](#_Toc190561071)

[2. Overall description 6](#_Toc190561072)

[2.1. Product Perspective 6](#_Toc190561073)

[2.2. Product Function 6](#_Toc190561074)

[2.3. User Classes and Characteristics 6](#_Toc190561075)

[2.4. User needs 7](#_Toc190561076)

[2.5. Assumption and Dependencies 7](#_Toc190561077)

[3. System features and requirements 9](#_Toc190561078)

[3.1. Functional Requirements 9](#_Toc190561079)

[3.2. External interface 9](#_Toc190561080)

[3.2.1. User Interface 9](#_Toc190561081)

[3.2.2. Hardware interface 9](#_Toc190561082)

[3.2.3. Software Interface 9](#_Toc190561083)

[4. Non-functional requirements 9](#_Toc190561084)

[4.1. Performance 9](#_Toc190561085)

[4.2. Security 9](#_Toc190561086)

[4.3. Reliability 9](#_Toc190561087)

**Acknowledgment**

# Introduction

In today's fast-paced digital world, personal productivity and effective task management have become critical skills for individuals and teams alike. Having a to-do list may help you prioritize your work and personal tasks. This project is available in English and has been developed in Python.

## Document Purpose

This Software Requirements Specification (SRS) document provides a detailed description of the requirements for the To-do list WebApp, which is developed as part of our semi-project. As the sole developer of this project, we are responsible for the entire process, from planning and design to implementation and testing.

This SRS outlines the functional and non-functional requirements, system constraints, and interface requirements that, to the final product, meet the desired objectives and satisfy both technical and user needs.

## Intended Use and Target Audience

This document is intended for the following audiences:

* Primary Users: Individuals and professionals seeking to organize their daily tasks and manage their time effectively
* Secondary Users: Small teams and study groups who need to track shared tasks
* Stakeholders: Developers, project managers, and system administrators who will maintain the application

## Product Scope

## Definitions, Acronyms and Abbreviations

**Definitions**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Administrator | System users with privileged access rights |
| Backend | Server-side application built with Django |
| Category | A grouping mechanism for related tasks |
| Database | Central storage system for all system data |
| Frontend | User-facing interface built with React |
| Priority | The level of importance assigned to a task (Low, Medium, High) |
| Task | A single unit of work or activity that needs to be completed |
| User | Any person who interacts with the system |

**Acronyms and Abbreviations**

|  |  |
| --- | --- |
| **Acronym** | **Full form** |
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |
| DB | Database |
| SRS | Software Requirements Specification |
| UI | User Interface |

## References

1. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
2. ISO 9001:2015 – Quality Management Systems
3. React.js Official Documentation
4. Django REST Framework Documentation
5. Python Official Documentation
6. Modern Web Development Best Practices Guide

# Overall description

## Product Perspective

The Todo List Web Application is a standalone web-based system that consists of:

1. **Frontend Component**
   * Built using React.js framework
   * Provides interactive user interface
   * Communicates with the backend through RESTful API
2. **Backend Component**
   * Developed using Django framework
   * Implements RESTful API endpoints
   * Manages data persistence and business logic
3. **Database Component**
   * Stores all task and user-related data
   * Ensures data consistency and reliability

The system is designed to be self-contained and does not depend on other external systems except for standard web infrastructure

## Product Function

The primary functions of the system include:

1. Create new tasks with title, description, status, due date, and category.
2. Update existing task details
3. Delete unwanted tasks

## User Classes and Characteristics

1. **Regular Users**

* Individuals managing personal tasks
* Frequency of use: Daily
* Technical expertise: Basic computer and internet skills
* Primary device: Desktop/laptop computers and mobile devices

1. **Student Users**

* Students managing academic tasks and deadlines
* Frequency of use: Multiple times per day
* Technical expertise: Intermediate
* Primary needs: Deadline tracking and task prioritization

1. **Professional Users**

* Business professionals managing work-related tasks
* Frequency of use: Throughout the workday
* Technical expertise: Intermediate to Advanced
* Primary needs: Task organization and priority management

## User needs

1. **Task Creation and Management**

* Quick and easy task creation
* Flexible task editing capabilities
* Clear task status visualization
* Efficient task deletion process

1. **Organization and Categorization**

* Intuitive category management
* Easy priority assignment
* Clear due date setting
* Simple status updates

1. **User Interface**

* Clean and intuitive design
* Responsive layout for all devices
* Easy navigation
* Clear visual feedback

## Assumption and Dependencies

1. **Technical Assumptions**

* Users have access to modern web browsers
* Users have stable internet connectivity
* Users have basic familiarity with web applications
* Devices meet minimum system requirements

1. **Dependencies**

* React.js framework for frontend development
* Django framework for backend development
* Database system for data storage
* Modern web browsers with JavaScript enabled
* Web server for hosting the application

1. **Environmental Dependencies**

* Compatible operating system
* Sufficient server resources
* Adequate network bandwidth
* Regular system maintenance and updates

# System features and requirements

## Functional requirements

(a). FR-1: Create Task

* **Title:** Create a New Task
* **Description:**  
  The system shall allow users to create a new task by providing details such as title, description, status, priority, due date, category, and an importance flag.
* **Postconditions**:
* A new task is successfully created and stored in the system.
* The task appears in the user's task list
* **Functional Requirements:**
  + - 1. **The system shall provide a form where users can enter task details, including:**
* Task title (string, required)
* Task description (string, optional)
* Task status (Default: “Pending”; Selected from “Pending,” “In Progress”)
* Due date (date required; must be today or later)
* Category (selected from a predefined list; required)
* Priority (selected from “Low”, “Medium”, “High”; required)
* Is Important (boolean, optional)
  + - 1. **The system shall validate user input before submitting**
* Title must not be empty.
* Due dates must not be in the past.
* A category must be selected.
  + - 1. **Upon successful creation, the system shall:**
* Store the task in the database.
* Display a confirmation message.
* Refresh the task list to include the newly created task.
  + - 1. **If task creation fails, the system shall display appropriate error messages, including:**
* "Please select a due date for the task."
* "Due date cannot be in the past."
* "Please select a category for the task."
* "Server error, please try again." (if applicable)

(b). FR-2: Update Task

* **Title:** Update task
* **Description:**  
  The system shall allow users to update an existing task by providing a form pre-populated with the task's current details. Users can modify the task's title, description, status, priority, due date, category, and importance flag.
* **Postconditions**:
* The task details will be updated in the system and reflected in the task list.
* The user will receive confirmation of the successful update.
* If an error occurs, an appropriate error message will be displayed.
* **Functional Requirements:**
  + - 1. **The system shall provide an editable form pre-populated with the current task details, including:**
* Task title (string, required)
* Task description (string, optional)
* Task status (Default: “Pending”; Selected from “Pending,” “In Progress”)
* Due date (date required; must be today or later)
* Category (selected from a predefined list; required)
* Priority (selected from “Low”, “Medium”, “High”; required)
* Is Important (boolean, optional)
  + - 1. **The system shall validate user input before submitting**
* Title must not be empty.
* Due dates must not be in the past.
* A category must be selected.
  + - 1. **Upon successful update, the system shall:**
* Save the updated task details in the database.
* Display a confirmation message to the user.
* Refresh the task list to reflect the updated task information.
  + - 1. **If a task update fails, the system shall display appropriate error messages, including:**
* "Please select a due date for the task."
* "Due date cannot be in the past."
* "Please select a category for the task."
* "Server error, please try again." (if applicable)

(c). FR-3: Delete Task

* **Title:** Delete task
* **Description:**  
  The system shall allow users to delete an existing task. When the user selects the delete option for a task, the system shall remove the task from the database and update the task list accordingly.
* **Postconditions**:
* The selected task is permanently removed from the system..
* The task list displayed to the user is refreshed to reflect the deletion..
* **Functional Requirements:**
  + - 1. **The system shall provide a delete option for each task in the task list:**
* The delete option shall be accessible via a dedicated button (e.g., labeled "Delete") associated with each task.
  + - 1. **Upon user initiation of a task deletion, the system shall:**
* Title must not be empty.
* Due dates must not be in the past.
* A category must be selected.
  + - 1. **Upon successful creation, the system shall:**
* Store the task in the database.
* Display a confirmation message.
* Refresh the task list to include the newly created task.
  + - 1. **If a task update fails, the system shall display appropriate error messages, including:**
* "Please select a due date for the task."
* "Due date cannot be in the past."
* "Please select a category for the task."

"Server error, please try again." (if applicable

## External interface

### User interface

### Hardware interface

### Software interface

# Non-functional requirements

## Performance

## Security

## Reliability