**University of Transport Ho Chi Minh City**

**Faculty of Information Technology**



**Software Requirement**

**Specification**

**for**

**To-do list**

**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](#_heading=h.gjdgxs)

[1.](#_heading=h.1fob9te) Introduction 4

[1.1.](#_heading=h.3znysh7) Purpose 4

[1.2.](#_heading=h.2et92p0) Intended Audience and Reading Suggestions 4

[1.3.](#_heading=h.tyjcwt) Product Scope 4

[1.4.](#_heading=h.3dy6vkm) Definitions, Acronyms and Abbreviations 4

[1.5.](#_heading=h.1t3h5sf) References 5

[2.](#_heading=h.2s8eyo1) Overall description 6

[2.1.](#_heading=h.17dp8vu) Product Perspective 6

[2.2.](#_heading=h.3rdcrjn) Product Function 6

[2.3.](#_heading=h.26in1rg) User Classes and Characteristics 6

[2.4.](#_heading=h.lnxbz9) User needs 6

[2.5.](#_heading=h.35nkun2) Assumptions and Dependencies 7

[3.](#_heading=h.1ksv4uv) Specific Requirements 7

[3.1.](#_heading=h.44sinio) Functional requirements 7

[3.2.](#_heading=h.2jxsxqh) External Interface Requirements 7

[3.2.1. User Interfaces 7](#_heading=h.z337ya)

[3.2.2. Hardware Interfaces 7](#_heading=h.3j2qqm3)

[3.2.3. Software Interfaces 7](#_heading=h.1y810tw)

[3.2.4. Communication Interfaces 7](#_heading=h.4i7ojhp)

[3.3.](#_heading=h.2xcytpi) System Features 7

[4.](#_heading=h.1ci93xb) Other Non-functional Requirements 7

[4.1.](#_heading=h.3whwml4) Performance Requirements 7

[4.2.](#_heading=h.2bn6wsx) Safety and Security Requirements 7

[4.3.](#_heading=h.qsh70q) Maintainability 7

[4.4.](#_heading=h.3as4poj) Portability 7

**Acknowledgement**

1. **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.

* 1. **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 oulines the functional and non-functional requirements, system constraints, and interface requirements that to the final product meets the desired objectives and satisfies both technical and user needs.

* 1. **Intended Audience and Reading Suggestions**

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.
  1. **Product Scope**

This To-do list WebApp is designed to

* 1. **Definitions, Acronyms and Abbreviations**

**Definitions**

| **Term** | **Definition** |
| --- | --- |
| Administrator | System user with priviledge 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 |

* 1. **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

1. **Overall description**
   1. **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 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.

* 1. **Product Function**

The primary functions of the system include:

1. **Task Management**

* Create new tasks with title, description, and due date
* Update existing task details
* Delete unwanted tasks
* Mark tasks as complete/incomplete
  1. **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 workday
* Technical expertise: Intermediate to Advanced
* Primary needs: Task organization and priority management
  1. **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
  1. **Assumptions and Dependencies**
  2. **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

1. **Specific Requirements**
   1. **Functional requirements**
   2. **External Interface Requirements**
      1. **User Interfaces**
      2. **Hardware Interfaces**
      3. **Software Interfaces**
      4. **Communication Interfaces**
   3. **System Features**
2. **Other Non-functional Requirements**
   1. **Performance Requirements**
   2. **Safety and Security Requirements**
   3. **Maintainability**
   4. **Portability**