Software Requirements Specification (SRS) for DayStride

1. Introduction

1.1 Purpose

DayStride is a web application for habit, goal, and to-do tracking, enabling users to build daily routines, track progress, and manage productivity effectively.

1.2 Scope

The system provides:

- Habit tracking with logs and completion history
- Goal management with public/private visibility and joinable goals
- To-do management with due dates
- Public GoalHub for discovering public goals
- Secure JWT-based user authentication and API access
- Web-friendly frontend

2. Functional Requirements

2.1 User Registration and Authentication

- Users can register with email and password.
- Users can log in and receive JWT access and refresh tokens.
- Users can log out, invalidating refresh tokens.

2.2 Habit Management

- Users can create, edit, delete, and view habits.
- Users can mark habits as completed for specific dates.
- System tracks and displays habit logs.
- Users can view habit completion history.

2.3 Goal Management

- Users can create, edit, delete goals.
- Goals can be set as public or private.
- Users can join other users' public goals.
- Joined goals do not appear in the GoalHub for that user.

2.4 GoalHub

- Displays all public goals from other users.
- Excludes user's own public goals and joined goals from their GoalHub view.
- Allows browsing and joining of public goals.
- Allows leaving public goals

2.5 To-Do Management

- Users can create, edit, delete to-do items.
- Users can mark to-do items as completed or pending.
- Supports due dates and times for to-do items.

2.6 Notifications

- Sends notification for addition of a habit/ todo/goal and join a goal.
- Notifications include the task name and due time.

2.7 Date and Time Format

- Dates: Jun 26 2026 (month (short), day, year)
- Times: 24-hour format (e.g., 17:30)

2.8 User Dashboard

Displays:

- Today's habits with completion toggles
- Active goals
- Today's to-do list
- Habit streak logs

2.9 User Activities

- Manage habits, goals, and to-dos . Joining and leaving public goals

3. Non-Functional Requirements

3.1 Performance

- Supports 1,000 concurrent users
- Page loads under 2 seconds during normal conditions

3.2 Scalability

The system shall be deployed using Kubernetes on AWS EC2, maintaining 5 pods each for the frontend, backend, and database services to support horizontal scaling and load distribution. The system shall use a scalable MySQL database with persistent volumes to ensure data consistency and high availability under increased load.

3.3 Security

- Passwords hashed with Django's password hasher
- JWT tokens for authentication
- HTTP enforced
- CORS restricted to trusted origins
- Passwords require lowercase, uppercase, numeric, and special characters

3.4 Availability

- Target uptime: 99.5% monthly

- Daily backups

3.5 Usability

- User-friendly forms with validation:
  - Task name: not blank, no special characters, not numbers only
  - Category: same rules as task name
  - Description: not blank
  - Due date: not earlier than the current day

3.6 Maintainability

- Clear backend/frontend separation
- Automated CI/CD with test, lint, build, and deploy pipelines

4. System Architecture

- **Frontend:** React SPA with REST API integration
- **Backend:** Django REST Framework with JWT
- **Database:** MySQL, managed with Django ORM
- **Deployment:** AWS EC2 with Docker Compose
- **Notifications:** SMTP for to-do reminders
- **CI/CD:** GitHub Actions