Daily Basis — Project Documentation

1. Project Title

Daily Basis
A subscription-based service / app for delivery of everyday essentials regularly, tailored to each user's schedule.

---

2. Overview

Daily Basis is a web/mobile application that delivers essential items people need daily — such as milk, bread, newspaper, eggs, curd, flowers, etc. — to users' doorsteps at scheduled times. Users subscribe (weekly or monthly), select the items they need on a recurring basis, and receive them daily. The system also allows changes (adding or removing items) for particular days, to accommodate varying needs.

---

3. Motivation / Problem Statement

- Everyday essentials are needed regularly; many people find it a hassle to purchase them daily.
- Current services (food delivery, ecommerce) are focused on on-demand or occasional purchases, not recurring essential items.
- Some essentials need to be delivered at fixed times (morning/evening), which many services do not guarantee.
- Users may forget to buy something or it may be inconvenient (time constraints, traffic, etc.).

---

4. Objectives

- To provide convenience by automating the delivery of daily essentials.
- To allow users to plan in advance and ensure they never miss important items.
- To reduce the daily purchase hassle and time spent shopping.
- To maintain flexibility, enabling users to modify their subscriptions as needed.
- To ensure reliable delivery at scheduled times (morning/evening).

---

5. Key Features

Feature	Description
User Registration & Authentication	Users can create accounts, log in securely.

Feature	Description
Plan Selection (Daily Basis)	Choose items needed every day (milk, newspaper, bakery, flowers, etc.).
Subscription Modes	Weekly or monthly subscription payment in advance.
Scheduled Delivery	Delivery at user-selected times (morning / evening), recurring daily.
Dynamic Adjustment	Users can add/remove items for specific days; request extras.
Payment Processing	Integration with payment gateway (examples: credit card, UPI, etc.).
User Dashboard	View upcoming deliveries, history, modify plans.
Notifications / Reminders	Remind user of upcoming delivery, confirm changes.

---

6. Technologies Used

- **Backend**: Python (Django) — server logic, database (from your GitHub structure I see manage.py, db.sqlite3). GitHub
- **Frontend / Templates**: HTML / CSS in templates directory. GitHub
- **Database**: SQLite (for development) — stores users, subscription details, item selections etc. GitHub
- **Static / Media Files**: For images, CSS / JS assets. GitHub
- **Deployment**: (if applicable) – note how you host / plan to host (server, cloud).

---

7. System Architecture

- **User Interface (UI)**: Web or Mobile front-end where user signs up, chooses plan, views dashboard, modifies subscription.
- **API / Backend Server**: Handles requests (registration, item selection, subscription management, changing items for days, etc.).
- **Database**: Stores user data, item catalogue, subscription schedules, delivery schedules, payments.
- **Delivery Scheduling Module**: Matches user schedules with delivery slots, ensures logistics support.
- **Payment Module**: Handles advance payments, recurring billing if needed.
- **Notifications Module**: Sends reminders, alerts or confirmation messages.

---

8. Workflow / Use Case Scenarios

1. **User Onboarding**
    - User downloads the app or visits the website.
    - Registers with email/phone.
    - Verifies identity if needed.
2. **Plan Setup**
    - User selects essential items (milk, newspaper, bakery items, etc.).
    - Chooses the delivery time (morning / evening).
    - Picks frequency (daily recurring) and subscription period (weekly / monthly).
    - Pays in advance for the subscription.
3. **Daily Delivery**
    - System triggers delivery each day at scheduled time.
    - Delivery staff / logistics picks up from vendors.
    - Items delivered to user address.
4. **Modifying the Plan**
    - If on a specific day user doesn't want an item (say bakery item), can exclude it.
    - If user wants extra for a day (e.g. extra milk or flowers), adds them.
    - Changes reflected for that day's delivery.
5. **Subscription Renewal**
    - After current period (e.g. a month) subscription ends or auto-renews (depending on policy).
    - User gets reminder to renew if manual.
6. **Payment / Billing**
    - Payments collected in advance.
    - If user makes modifications, pro-rating or extra charges handled.

---

9. Database Schema (High-level)

- **User**: id, name, contact details, address, etc.
- **Item**: id, name, category (milk, bakery, newspaper, etc.), price.
- **SubscriptionPlan**: id, user_id, start_date, end_date, delivery_time, frequency.
- **SubscriptionItem**: link between SubscriptionPlan and Item (with quantity).

- **DailyOverride**: for specific days: additions/removals or extra items.
- **Payment**: id, user_id, subscription_id, amount, date, status.
- **DeliverySchedule**: schedule for deliveries per user per day.

---

10. UI / Screens (Suggested / Implemented)

- **Login / Registration Screen**
- **Plan Creation / Item Selection Screen**
- **Dashboard**: show upcoming days' items, next delivery, past deliveries.
- **Modify Plan / Override Day Screen**
- **Payment Screen**
- **Profile / Settings**

(Add any screenshots you have here.)

---

11. Challenges & Solutions

Challenge	How We Addressed It (or Plan to)
Ensuring daily timely delivery (morning/evening)	Create fixed delivery slots, partner with local vendors & delivery agents.
Handling changes (adding/removing items) without confusion	Maintain day-wise overrides; keep user notifications; dynamic dashboard.
Managing inventory of vendors / supply chain	Sync with vendors periodically; have buffer inventory for high demand.
Handling missed days / payment failures	Reminders; cancellation policies; fallback plan.
Scalability	Starting small with limited areas, then scaling vendor network, delivery staff, infrastructure.

---

12. Future Enhancements

- Expand catalog of daily essentials (e.g. medicines, fruits, vegetables).
- Implement mobile app version (iOS / Android) for better UX.
- Introduce auto-renewal, loyalty rewards.
- Add more flexible delivery windows (multiple time slots).
- Implement real-time tracking of delivery.

- Support multiple payment options (wallets, UPI, cards).
- Add feedback / rating system for items and delivery.

---

13. Deployment & Maintenance

- **Hosting**: Where the backend and front-end are hosted (cloud, VPS etc.).
- **Version Control**: Git / GitHub (current repo "hackhathon") for source control. GitHub
- **Testing**: Unit testing for important modules (subscription, billing), user acceptance testing.
- **Monitoring & Support**: Logs, performance monitoring, customer support channels.
- **Data Backup & Security**: Secure user data, payment info; routine backups of database.

---

14. Results & Learnings

- What we achieved: built a working prototype (web app) with basic subscription, item selection, daily schedule.
- What we couldn't do / scope limitations (during hackathon): maybe delivery partner integration, real-payment gateway, mobile app etc.
- Learnings: importance of UX for recurring tasks; challenges of handling per-day modifications; database design for overrides.

---

15. Conclusion

Daily Basis seeks to transform how people manage their daily essentials: making the process automatic, reliable, and flexible. The success of such a system lies in its ability to understand user routines, adjust to day-to-day changes, and deliver consistently. With further development and expansion, Daily Basis has the potential to become a very useful service in urban and semi-urban settings, reducing friction in daily life.