**MealMate - Project Specification**

**Product Owner for Sprint 1: Ria Wang (Andrew ID: yerouw)**

**1. Product Backlog**

The following features define the complete functionality of MealMate. These are grouped into key modules.

**1.1 User Authentication & Profiles**

* **User Registration & Login** (Django Auth/OAuth)
  + Users can sign up/log in using email and password.
  + OAuth-based authentication (Google login) may be added if time permits.
* **User Profiles**
  + Users can update profile details (name, contact info, profile picture).
  + Users can view other participants’ profiles in events.

**1.2 Meal Event Management**

* **Create Dining Event**
  + Users can create meal events specifying:
    - Event Name
    - Time & Date
    - Location (Google Maps API)
    - Max Participants
    - Privacy (Public/Invite-Only)
* **Join Meal Event**
  + Users can browse & join available events.
  + Event creators can manage attendance (remove users, close event).
* **Location-based Event Recommendations**
  + Nearby meal events are suggested based on the user’s location.
  + Sorting based on proximity and popularity.

**1.3 Bill Splitting & Payment**

* **Equal or Custom Bill Splitting**
  + Organizers can enable an “AA Split” (Equal Cost Sharing) mode.
  + Users can specify custom amounts if needed.
* **Payment Integration (Stripe/PayPal API)**
  + Users can make payments within the app.
  + Payment status (Pending/Completed) is tracked.

**1.4 Real-Time Group Chat**

* **WebSocket-powered chatrooms**
  + Users are automatically added to a chatroom after joining an event.
  + Participants can discuss logistics, confirm attendance, and chat before the meal.

**Task Allocation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Description** | **Tasks** | **Deliverables** | **Assigned To** |
| User Registration & Login | Implement Django authentication system, allowing users to sign up and log in securely. | - Set up Django authentication model - Create API endpoints for registration/login - Implement session-based authentication | - Working registration/login API - User authentication flow tested | Lu |
| Create Meal Event | Users can create meal events by specifying details such as location, time, and participants. | - Design event creation model - Build API for event creation - Validate event form fields | - API for creating events - Basic event creation UI | Ria |
| Join Event | Users can browse available events and join them. | - Implement API for joining events - Update event participant count - UI for browsing & joining events | - API for joining events - Event list page | Ria |
| Frontend Page Layout | Implement homepage and event details page UI. | - Design basic UI with placeholders - Set up navigation and routing - Display event details | - Homepage & event details page | Ria |
| Database Setup | Define Django models for users & events. | - Create models for User, Event, Profile - Migrate database and test schema | - models.py file - Working database schema | Lu |
| User Location Handling | Store user’s current location (manually set or auto-detected). | - Add location fields in User model - Implement API for updating user location - Allow users to set location manually | - API for storing user location - Profile location settings | Lu |
| Google Maps API Integration | Display meal event locations on an interactive map. | - Set up Google Maps API / OpenStreetMap - Show meal events on the map - Add clickable markers for event details | - Interactive event map - Location-based search | Lu |
| Profile Management | Allow users to update their profile details, including name, picture, and dietary preferences. | - Create Django model for user profile - Implement profile update API - Add UI for profile editing | - Profile update page - Backend API for updating profile | Lu |
| Nearby Event Recommendations | Recommend meal events based on user location. | - Implement geolocation-based filtering - Fetch and display nearest events - Optimize query performance | - 'Nearby Events' section in homepage - Location-based search API | Ria |
| Bill Splitting & Payments | Allow event organizers to enable bill splitting and let users pay online or manually. | - Implement payment model - Integrate Stripe/PayPal API - Allow users to manually mark payments | - Bill splitting UI & backend - Payment transaction records | Ria |
| Payment Status Updates | Update users' payment status in real-time when they complete transactions. | - Use WebSocket to update payment status - Notify users when a payment is completed - Display outstanding balances | - Live payment updates via WebSocket - Payment confirmation notifications | Ria |

**2. First Sprint Backlog**

For the first sprint, we focus on **core functionalities** while keeping the workload reasonable. **Live chat and payments will not be implemented in this sprint.** Instead, we focus on **user authentication, event creation/joining, and data modeling validation.**

**Sprint 1 Deliverables**

**2.1 Backend Setup & Database Design**

* **Django Backend Setup**
  + Initialize the project with Django.
  + Set up basic project structure.
* **Database Schema & ER Diagram Finalization**
  + **ER diagram validation before implementing models** (to prevent rework).
  + Once confirmed, implement core models in Django.

**2.2 Frontend Setup**

**2.3 User Authentication & Profiles**

* **User Registration & Login**
  + Implement Django authentication.
  + Create sign-up and login views.
* **User Profile Management**
  + Implement basic profile editing.

**2.4 Meal Event Management**

* **Create Meal Event (Basic)**
  + Implement event creation with form validation.
  + Store event details in the database.
* **Join Meal Event**
  + Implement event joining functionality.

**Deferred to Future Sprints**

* **Meal Event Management**
* **Bill splitting & payments**
* **OAuth integration (Google login)**
* **Event recommendations (location-based)**
* **Live chat (WebSocket)**

**Task Allocation**

* **Ria Wang:** Front-End Page Build & Design, Event Details, Meal Creation, Event Joining
* **Xiangning Lu:** Database Setup/ER Diagram Refinement, User Registration & Login

**3. Data Models (Work-in-Progress)**

**As models are still being refined, this section is subject to updates after ER diagram validation.** Below is a rough version:

**User Model**

class User(models.Model):

email = models.EmailField(unique=True)

password = models.CharField(max\_length=255)

profile\_picture = models.ImageField(upload\_to="profiles/", blank=True, null=True)

join\_date = models.DateTimeField(auto\_now\_add=True)

**Event Model**

class Event(models.Model):

title = models.CharField(max\_length=100)

host = models.ForeignKey(User, on\_delete=models.CASCADE)

location = models.CharField(max\_length=255)

date\_time = models.DateTimeField()

max\_participants = models.IntegerField()

participants = models.ManyToManyField(User, related\_name="joined\_events")

status = models.BooleanField(default=True) # Active or Canceled

**Payment Model (Future Sprint)**

*Not implemented in Sprint 1.*