**Functional Requirements**

1. **User Authentication**:
   * Students can register with an email and password.
   * Students can log in securely to access their wallet.
   * Option to log out.
2. **Expense Tracking**:
   * Students can add expenses with details like amount, category (e.g., Food, Books), and date.
   * Students can view a list of all past expenses.
   * Students can edit or delete existing expenses.
3. **Spending Charts**:
   * Display spending data in charts (pie or bar) for daily, weekly, and monthly periods.
   * Filter charts by category or time range.
   * Show total spending for the selected period.
4. **Budget Setting**:
   * Students can set budget limits for specific categories (e.g., $50 for Food monthly).
   * Display warnings when spending nears or exceeds budget limits.
   * Allow editing or resetting budgets.
5. **Bank Connectivity (Future Feature)**:
   * Enable seamless transfers from students’ bank accounts to the app’s wallet.
   * Display transfer history.
6. **Savings Model with Interest (Future Feature)**:
   * Allow students to allocate money to a savings section.
   * Calculate and display interest earned (e.g., fixed rate like 2% annually).
   * Option to withdraw savings back to the wallet.

**Non-Functional Requirements**

1. **Usability**:
   * Interface must be intuitive and easy to navigate for students with minimal tech experience.
   * Load times for screens and charts should be under 2 seconds.
   * Design should be visually appealing and consistent.
2. **Security**:
   * User data (e.g., email, expenses) must be encrypted during storage and transmission.
   * Authentication credentials must be securely handled.
   * Prevent unauthorized access to user accounts.
3. **Performance**:
   * App should handle up to 1,000 transactions per user without slowdown.
   * Charts should render smoothly even with large datasets.
   * Minimal battery and data usage.
4. **Reliability**:
   * App should have 99% uptime for cloud-based features (if using a backend).
   * Handle errors gracefully with clear messages (e.g., “Insufficient balance”).
   * Data should be backed up to prevent loss.
5. **Scalability**:
   * Support at least 10,000 concurrent users if using a backend.
   * Allow for future feature additions like bank APIs or notifications.
6. **Compatibility**:
   * Run on Android devices with API level 21 (Lollipop) or higher.
   * Support various screen sizes and resolutions.