|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | User Story | Estimate | Acceptance Criteria | Task(s) |
| 1 | As a user, I want to be able to create a username and password to protect my data. | S | 1. Each username is unique. 2. Each username has a recovery question and answer. 3. Password must contain at least one number. 4. If username and password are correct login succeeds. 5. An annual income amount is entered. 6. A desired savings amount is entered. | 1. Design user table. 2. Create user table. 3. Design UI to create user. 4. Test database connectivity. 5. Test recovery question. |
| 2 | As a user, I want to be able to store a budget weekly | S | 1. Check user exist. 2. Budget is assigned to a user. 3. A disposable income amount is entered for calculations. 4. Budget is calculated for one week with even spending each day. 5. The budget knows which week it belongs to. | 1. Design budget table 2. Create budget table 3. Design UI to create weekly budget. 4. Test database connectivity. |
| 3 | As a user, I want to be able to store a budget monthly. | M | 1. Check user exist. 2. Budget is assigned to a user. 3. A disposable income amount is entered for calculations. 4. Budget is calculated for one month with even spending each day. 5. The budget knows which month it belongs to. | 1. Design budget table 2. Create budget table 3. Design UI to create monthly budget. 4. Test database connectivity. |
| 4 | As a user, I want to be able to store a budget yearly. | L | 1. Check user exist. 2. Budget is assigned to a user. 3. A disposable income amount is entered for calculations. 4. Budget is calculated for one year with even spending each day.   The budget knows which Year. It belongs to. | 1. Design budget table 2. Create budget table 3. Design UI to create yearly budget. 4. Test database connectivity |
| 5 | As a user, I want to be able to add items to the budget. | S | 1. Item contains only characters. 2. Duplicate items are stored as one item with an amount. 3. Item must belong to a budget. | 1. Design items table. 2. Create items table. 3. Design UI to add items button on main page. 4. Design UI to add item to budget. 5. Code the items UI. 6. Test UI. 7. Test database connectivity of Items table. |
| 6 | As a user, I want to track items by cost. | S | 1. Item must have a price. 2. Price cannot be negative. 3. Price is assumed to be the same currency as the budget. | 1. Design items table. 2. Create items table. 3. Test entering a negative price. |
| 7 | As a user, I want to track items by date. | S | 1. Every item must have a date. 2. Date must be valid. 3. Item is assigned a date. 4. Future dates must be rejected. | 1. Design items table. 2. Create items table. 3. Create date format to be entered. 4. Test dates that aren’t in the right format. 5. Test invalid future dates. |
| 8 | As a user, I want to track items by type. | M | 1. Type must be a valid type in the types table. 2. Item must have a type/category. | 1. Design a types table. 2. Create types table. 3. Test to see if incorrect types get through. |
| 9 | As a user, I want to track items by amount. | S | 1. Every item must have an amount. 2. Items must be able to be sorted by amount. 3. Duplicate items increase the amount of an item. | 1. Design a items table. 2. Create a items table. 3. Code UI for amount in items entry UI 4. Test entering duplicate items. |
| 10 | As a user, I want a way to display all items towards a budget. | M | 1. Every item in a budget is displayed. 2. Sortable by price, type, amount, name, and date. | 1. Design display UI. 2. Create Item display UI. 3. Test display function. 4. Test name sort. 5. Test price sort. 6. Test amount sort. 7. Test date sort. |