METER BOX APP PHP Application Development Progress

(80% Completion and System Documentation)

1. Functional Requirements Checklist

Core Features Checklist
The table below highlights the core features of the application, their status, and a brief description of their functionality.

Feature	**Status**	**Description**
User Authentication	✔️ Completed	Allows users to register, login, and manage their accounts securely (PHP).
CRUD Operations for Tokens	✔️ Completed	Users can Create, Read, Update, and Delete tokens related to their meter box.
API Integration	✔️ Completed	Integrated with the City of Tshwane API to validate meter box numbers.
Token Purchase Functionality	✔️ Completed	Users can purchase electricity and view the tokens generated post-purchase.
Frontend Functionality	✔️ Completed	The PHP-based frontend includes forms and interfaces for token management.
Backend Functionality	✔️ Completed	Backend PHP code handles form submissions, database queries, and logic.
Responsive Design	✔️ Completed	The app is responsive across mobile, tablet, and desktop devices.
Manual vs Electronic Delivery	✔️ Completed	Users can choose between manual token entry or electronic transmission.
Real-Time Feedback	❌ In Progress	Users will receive real-time feedback on token status.
Live Support Chat Integration	❌ In Progress	Chatbot support feature under development.

Summary
As of this report, **80%** of the required features are functional. The app allows users to authenticate, purchase electricity, and manage tokens via both manual and electronic transmission methods. Responsive design is implemented for optimal user experience on various devices.

2. System Documentation

2.1 Technical Architecture

- Frontend: PHP with HTML/CSS and JavaScript for building the user interface.
- Backend: PHP handles server-side logic, user requests, and interactions with the database.
- Database: MySQL stores user details, meter box data, and purchase records.
- API: Integration with the City of Tshwane API to validate meter boxes and send tokens.
- Communication Flow: User actions trigger frontend requests processed by PHP, which interacts with MySQL and external APIs. The data is then rendered on the frontend.

2.2 Database Schema

The database schema focuses on user and token management. Below is a simplified structure:

Tables:

- Users Table

Field Name	Type	Description
user_id	INT (Primary)	Unique identifier for each user.
name	VARCHAR	Name of the user.
email	VARCHAR	Email for login and communication.
password	VARCHAR	Hashed password for security.
created_at	TIMESTAMP	Date of account creation.

- Tokens Table

Field Name	Type	Description
token_id	INT (Primary)	Unique token identifier.
user_id	INT (Foreign)	Associated user who purchased the token.
meter_box_no	VARCHAR	User's 11-digit meter box number.
amount	DECIMAL	Amount of electricity purchased.
token_code	VARCHAR	The token number received after purchase.
status	ENUM	Whether the token was manually entered or electronically transmitted.
created_at	TIMESTAMP	Timestamp of when the token was generated.

2.3 API Endpoints

Method	**Endpoint**	**Description**
`POST`	`/auth/register.php`	Registers a new user in the system.
`POST`	`/auth/login.php`	Authenticates the user and starts a session.

`GET`	`/tokens/list.php?user_id={user_id}`	Retrieves tokens associated with the user.
`POST`	`/tokens/purchase.php`	Handles the electricity token purchase process.
`PUT`	`/tokens/update.php?token_id={token_id}`	Updates token status (manual or electronic).

2.4 User Interface (UI) Design

Screens Overview:

1. **Login & Registration:**
   Simple, responsive forms for users to authenticate.

2. **Dashboard:**
   Displays the user's meter box number, purchase history, and options for new purchases.

3. **Token Purchase Page:**
   Users can input meter box numbers, select a bank, and purchase electricity. It also displays token details post-purchase.

4. **Responsive Design:**
   The app adapts seamlessly across desktop, tablet, and mobile devices, ensuring a smooth user experience.

2.5 Instructions for Running the Application

Development Setup:
1. Clone the repository from GitHub.
2. Ensure a local server (e.g., XAMPP or WAMP) is running.
3. Place the project folder in the htdocs directory.
4. Import the MySQL database schema using PHPMyAdmin.
5. Configure database credentials in the `connection.php` file.
6. Start the local server and access the app through the browser.
7. Run the necessary PHP files (login, registration, etc.) to test functionality.

Deployment:
1. Deploy the application on a web hosting service with PHP and MySQL support.
2. Configure database credentials on the live server.
3. Ensure that API keys for the City of Tshwane integration are properly set.
4. The app will be accessible via the domain URL.

3. Problem Statement

Access to electricity is a fundamental need in modern society. However, in certain regions, users often face difficulties in purchasing electricity tokens, especially during off-hours when physical purchase points are closed. Many users also experience delays in manually entering tokens into their meter boxes, leading to unnecessary power interruptions.

Khanyisa , our meter_box_app, addresses this issue by allowing users to easily purchase electricity tokens online. It provides a seamless experience where tokens can be automatically loaded into the meter box, reducing the inconvenience of manual input. This system not only enhances access to electricity but also reduces the time users spend managing their energy needs. By promoting more efficient energy management, the app contributes to better quality of life, especially in underserved communities where access to utilities is inconsistent.

Submission Requirements Checklist:

- [x] Functional Requirements Checklist
- [x] System Documentation (Technical Architecture, Database Schema, API Endpoints, UI Design, Instructions for Running the Application)
- [x] Problem Statement