PayHab API

API Version: 2.0.0
This is the PayHab Client API

By Dream Team

INDEX

1. **OVERVIEW**
    - ○ BASE URL
    - ○ AUTHENTICATION
    - ○ HEADERS
    - ○ PUBLIC ENDPOINTS
    - ○ RATE LIMITING
    - ○ ERROR RESPONSES
    - ○ RESPONSE STATUS CODES
    - ○ REQUIRED HEADERS
2. **USER AUTHENTICATION & MANAGEMENT**
    - ○ POST /register
    - ○ POST /login
    - ○ POST /forgot-password
    - ○ POST /verify-token
    - ○ GET /user/{id}
3. **TRANSACTION MANAGEMENT**
    - ○ GET /transactions/recent/{user_id}
4. **LOAN MANAGEMENT**
    - ○ POST /loans/add
    - ○ POST /loans/clear
    - ○ GET /loans/get
5. **EMAIL VERIFICATION**
    - ○ POST /send-verification

1. OVERVIEW

1.1 BASE URL

`http://10.0.2.2:5000`

1.2 AUTHENTICATION

Bearer token required for protected endpoints.

1.3 Headers:

All requests must include:

```
{
"Content-Type": "application/json",

}
```

Protected endpoints additionally require:

```
{
"Authorization": "Bearer {idToken}"
}
```

1.4 Public Endpoints:

- POST /register
- POST /login
- POST /forgot-password

1.5 RATE LIMITING

- Login attempts: 5 per minute per IP
- Failed login lockout: 300 seconds after 5 failed attempts

1.6 ERROR RESPONSES

All endpoints return errors in the following format:

```
{
  "error": "Error description"
}
```

1.7 Common HTTP Status Codes:

- 200: Success
- 400: Bad Request
- 401: Unauthorized
- 404: Not Found
- 429: Too Many Requests

---

2. USER AUTHENTICATION & MANAGEMENT

2.1 POST /register:

Purpose: Register a new student account.

Request:

```
{

  "name": "string",

  "studentId": "string",
  "email": "string",
  "password": "string"
}
```

Response:

200 OK:

```
{
  "message": "User created successfully."
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

2.2 POST /login

Purpose: Authenticate the student and return access tokens.

Request:

```
{
  "email": "string",
  "password": "string"
}
```

Response:

200 OK:

```
{
  "message": "Login successful",
  "idToken": "string",
  "refreshToken": "string",
  "userId": "string"
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

429 Too Many Requests:

```
{
  "error": "Account locked due to multiple failed attempts. Please try again later."
}
```

2.3 POST /forgot-password

Purpose: Send password reset email to user.

Request:

```
{
  "email": "string"
}
```

Response:

200 OK:

```
{
  "message": "Password reset email sent successfully"
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

2.4 POST /verify-token

Purpose: Verify if an ID token is valid.

Request:

```
{
  "idToken": "string"
}
```

Response:

200 OK:

```
{
  "message": "Token is valid"
}
```

401 Unauthorized:

```
{
  "error": "Token invalid"
}
```

2.5 GET /user/{id}

Purpose: Retrieve user profile details.

Response:

200 OK:

```
{
  "data": {
    "name": "string",
    "studentId": "string",
    "email": "string",
    "forceLogout": boolean
  }
}
```

404 Not Found:

```
{
  "error": "User not found"
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

---

3. TRANSACTION MANAGEMENT

3.1 GET /transactions/recent/{user_id}

Purpose: Retrieve recent transactions for a user (up to 10, ordered by most recent first).

Response:

200 OK:

```
{
  "transactions": [
    {
      "id": "string",
      "type": "loan_add/loan_clear",
      "vendor": "string",
      "amount": number,
      "timestamp": "ISO-8601 timestamp"
    }
  ]
}
```

404 Not Found:

```
{
  "error": "User not found"
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

---

4. LOAN MANAGEMENT

4.1 POST /loans/add

Purpose: Record a new loan amount for a vendor.

Request:

```
{
  "userId": "string",
  "vendor": "string",
  "amount": number
}
```

Response:

200 OK:

```
{
  "message": "Loan added successfully",
  "newAmount": number
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

4.2 POST /loans/clear

Purpose: Process a payment to clear a loan partially or completely.

Request:

```
{
  "userId": "string",
  "vendor": "string",
  "amount": number
}
```

Response:

200 OK:

```
{
  "message": "Loan cleared successfully",
  "newAmount": number
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

4.3 GET /loans/get

Purpose: Get the current loan amount for a specific vendor.

Request Parameters:

- userId (query parameter)
- vendor (query parameter)

Response:

200 OK:

```
{
  "amount": number
}
```

400 Bad Request:

```
{
  "error": "string"
}
```

---

5. EMAIL VERIFICATION

5.1 POST /send-verification

Purpose: Send an email verification to the user.

Request:

```
{
  "idToken": "string"
```

}

Response:

200 OK:

```
{
  "message": "Verification email sent successfully"
}
```

400 Bad Request:

```
{
  "error": "string"
}
```
