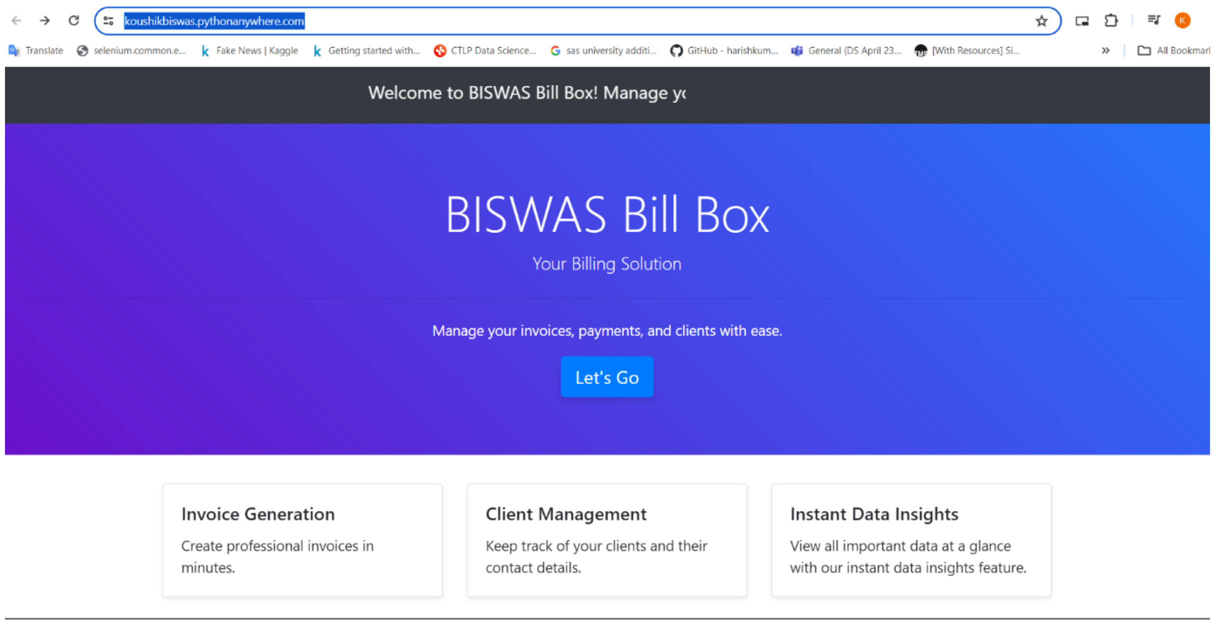
Biswas Bill Box Documentation

(Date:- 23/05/2024)

Overview

BISWAS Bill Box is a comprehensive billing application designed to manage user details, products, billing information, and invoice product details. This documentation provides an overview of the project structure, setup instructions, endpoints, models, functions, and usage of JavaScript and Bootstrap for front-end development.



Welcome to BISWAS Bill Box! Manage yc

BISWAS Bill Box

Your Billing Solution

Manage your invoices, payments, and clients with ease.

Invoice Generation

Create professional invoices in minutes.

Client Management

Keep track of your clients and their contact details.

Instant Data Insights

View all important data at a glance with our instant data insights feature.

Documentation sections:

- What is BISWAS Bill Box?
- How to Use BISWAS Bill Box: Step-by-Step Guide
- Available Endpoints
- Data Models
- User Registration
- Conclusion

<u>What is BISWAS Bill Box?</u>

BISWAS Bill Box is a user-friendly billing application designed to streamline the process of managing users, products, bills, and invoices. It offers a robust and intuitive interface for handling various billing operations, making it ideal for small businesses and individual entrepreneurs.

- Invoice Generation: Create professional invoices in minutes.
- Client Management: Keep track of your clients and their contact details.
- Transaction Management: Keep track of your transaction and the product details.
- Instant Data Insights: View all important data at a glance with our instant data insights feature.

How to Use BISWAS Bill Box: Step-by-Step Guide

To get started with BISWAS Bill Box, follow these steps:

1. **Clone the Repository**: Download or clone the project repository to your local machine. *Repository LInk*

2. **Navigate to the Project Directory**: Use your terminal or command prompt to go to the project root directory.

3. **Install Dependencies**: Install the necessary Python packages using **pip**.

   All the files are present in requirements.txt folder

4. **Configure the Database**: Set up your database and configure the connection settings.

   You can modify the database connection from settings.py file inside the app

5. **Run the Application**: Start the application using the appropriate command.

   open cmd and navigate to the project Directory and just write:

   ```
   python manage.py runserver
   ```

6. **Access the Application**: Open your web browser and navigate to the application's URL.
   URL:- http://127.0.0.1:8000/
7. **Register a New User**: Use the registration endpoint to create a new user account.
8. **Log In**: Log in with your credentials to access the dashboard.
9. **Add Products**: Use the interface to add new products to the system.
10. **Generate Invoices**: Create invoices for users and manage billing details after Add Products.

Available Endpoints:-

1. **Sign-In Page**
   - **URL**: `sign-in/`
   - **View**: `views.sign_in_page_view`
   - **Name**: `sign_in`
   - **Description**: Displays the sign-in page for users to enter their credentials.
2. **Welcome Page**
   - **URL**: `welcome/`
   - **View**: `views.sign_in_view`
   - **Name**: `process_email`
   - **Description**: Processes the user's sign-in information and redirects to the welcome page.
3. **User Details Input**
   - **URL**: `userdetails/`
   - **View**: `views.input_user_details_view`
   - **Name**: `user_address`

- **Description**: Displays a form for the user to input their details such as name, address, and contact information.

4. **Add Product**

- **URL**: `addproduct/<str:string1>/`
- **View**: `views.add_product_view`
- **Name**: `user_products`
- **Description**: Allows users to add a new product to their account. The `string1` parameter can be used to pass additional data to the view.

5. **View All Products**

- **URL**: `view_all_products/<str:string1>/`
- **View**: `views.view_all_products`
- **Name**: `view_all_products`
- **Description**: Displays a list of all products associated with the user. The `string1` parameter can be used to filter or categorize the products.

6. **Generate Invoice**

- **URL**: `generate-invoice/<str:mob_no>/`
- **View**: `views.generate_invoice_view`
- **Name**: `generate_invoice`
- **Description**: Generates a new invoice for the user identified by `mob_no`.

7. **View Invoice**

- **URL**: `invoice/<str:mob_no>/<str:invoice_no>/`
- **View**: `views.genarateInvoice`
- **Name**: `invoice`
- **Description**: Displays the details of a specific invoice identified by the user's mobile number (`mob_no`) and the invoice number (`invoice_no`).

8. **Customer Details**

- **URL**: `customer-details/<str:mob_no>/`
- **View**: `views.customer_details_view`

- **Name**: `customer_details`
- **Description**: Shows the details of the customer identified by `mob_no`, including contact information and billing history.

9. **Transaction Details**

- **URL**: `transaction-details/<str:mob_no>/`
- **View**: `views.transaction_details_view`
- **Name**: `transaction_details`
- **Description**: Displays the transaction history of the customer identified by `mob_no`.

10. **Reports**

- **URL**: `reports/<str:user_id>/`
- **View**: `reports.reports_view`
- **Name**: `reports_view`
- **Description**: Generates and displays reports for the user identified by `user_id`.

11. **Help Page**

- **URL**: `help/`
- **View**: `views.help_view`
- **Name**: `help_page`
- **Description**: Provides help and support information for users.

12. **Search Invoice**

- **URL**: `search-invoice/<str:user_id>/`
- **View**: `views.search_invoice_view`
- **Name**: `search_invoice`
- **Description**: Allows users to search for invoices using their user ID.

Data Models:- This section provides a detailed description of the data models used in the BISWAS Bill Box application. These models represent the core entities in the system, including users, products, bills, and invoice product details.

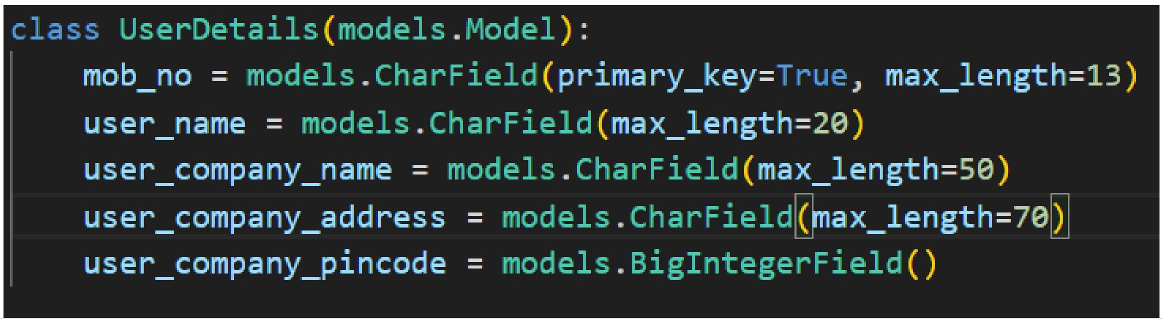
1. UserDetails
2. AddProduct
3. BillDetails
4. InvoiceProductDetails

• UserDetails

The `UserDetails` model stores information about the users of the BISWAS Bill Box application.

Fields:

- **mob_no**: A `CharField` serving as the primary key, representing the user's mobile number. Maximum length is 13 characters.
- **user_name**: A `CharField` for the user's name. Maximum length is 20 characters.
- **user_company_name**: A `CharField` for the user's company name. Maximum length is 50 characters.
- **user_company_address**: A `CharField` for the user's company address. Maximum length is 70 characters.
- **user_company_pincode**: A `BigIntegerField` for the user's company pincode.



```
class UserDetails(models.Model):
    mob_no = models.CharField(primary_key=True, max_length=13)
    user_name = models.CharField(max_length=20)
    user_company_name = models.CharField(max_length=50)
    user_company_address = models.CharField(max_length=70)
    user_company_pincode = models.BigIntegerField()
```

AddProduct

The `AddProduct` model stores information about products associated with a user.

Fields:

- **user**: A `ForeignKey` linking to `UserDetails`, with `CASCADE` deletion.
- **name**: A `CharField` for the product's name. Maximum length is 100 characters.
- **details**: A `TextField` for detailed information about the product.

- **quantity**: A `PositiveIntegerField` for the quantity of the product.
- **price**: A `DecimalField` for the product's price. Maximum of 10 digits, with 2 decimal places.
- **selling_price**: A `DecimalField` for the product's selling price. Maximum of 10 digits, with 2 decimal places.

```
class AddProduct(models.Model):
    user = models.ForeignKey(UserDetails, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    details = models.TextField()
    quantity = models.PositiveIntegerField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    selling_price = models.DecimalField(max_digits=10, decimal_places=2)
```

BillDetails

The `BillDetails` model stores information about bills generated by users.

Fields:

- **user**: A `ForeignKey` linking to `UserDetails`, with `CASCADE` deletion.
- **bill_number**: A `CharField` for the bill number. Maximum length is 6 characters.
- **bill_date**: A `DateField` for the date of the bill.
- **bill_time**: A `TimeField` for the time of the bill.
- **customer_name**: A `CharField` for the customer's name. Maximum length is 100 characters.
- **customer_mobile_no**: A `CharField` for the customer's mobile number. Maximum length is 15 characters.
- **customer_address**: A `CharField` for the customer's address. Maximum length is 200 characters.
- **bill_amount**: A `DecimalField` for the total amount of the bill. Maximum of 10 digits, with 2 decimal places.

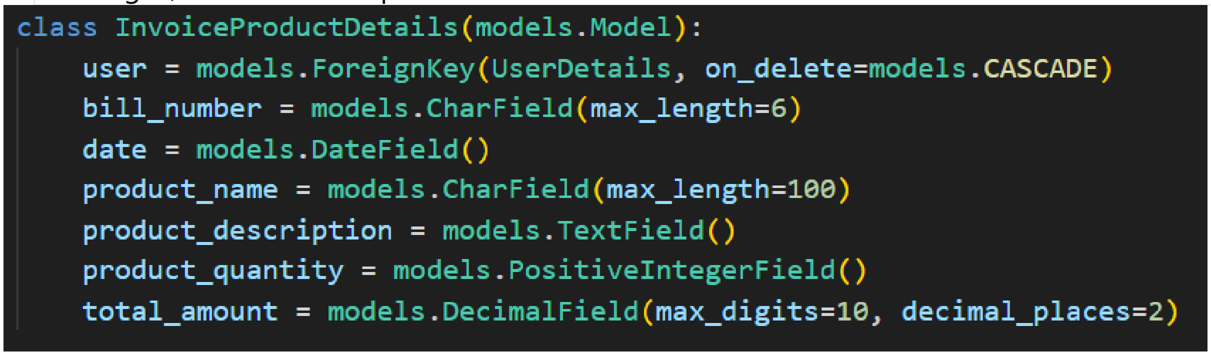
```
class BillDetails(models.Model):
    user = models.ForeignKey(UserDetails, on_delete=models.CASCADE)
    bill_number = models.CharField(max_length=6)
    bill_date = models.DateField()
    bill_time = models.TimeField()
    customer_name = models.CharField(max_length=100)
    customer_mobile_no = models.CharField(max_length=15)
    customer_address = models.CharField(max_length=200)
    bill_amount = models.DecimalField(max_digits=10, decimal_places=2)
```

InvoiceProductDetails

The `InvoiceProductDetails` model stores information about products included in invoices.

Fields:

- **user**: A `ForeignKey` linking to `UserDetails`, with `CASCADE` deletion.
- **bill_number**: A `CharField` for the bill number. Maximum length is 6 characters.
- **date**: A `DateField` for the date of the invoice.
- **product_name**: A `CharField` for the product name. Maximum length is 100 characters.
- **product_description**: A `TextField` for the product description.
- **product_quantity**: A `PositiveIntegerField` for the quantity of the product.
- **total_amount**: A `DecimalField` for the total amount of the product. Maximum of 10 digits, with 2 decimal places.



```python
class InvoiceProductDetails(models.Model):
    user = models.ForeignKey(UserDetails, on_delete=models.CASCADE)
    bill_number = models.CharField(max_length=6)
    date = models.DateField()
    product_name = models.CharField(max_length=100)
    product_description = models.TextField()
    product_quantity = models.PositiveIntegerField()
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)
```

User Registration: -

Step-by-Step Instructions

Step 1: Click on the "Let’s Go" Icon

Begin by clicking on the "Let’s Go" icon to start using BISWAS Bill Box.

Step 2: Enter Your Mobile Number

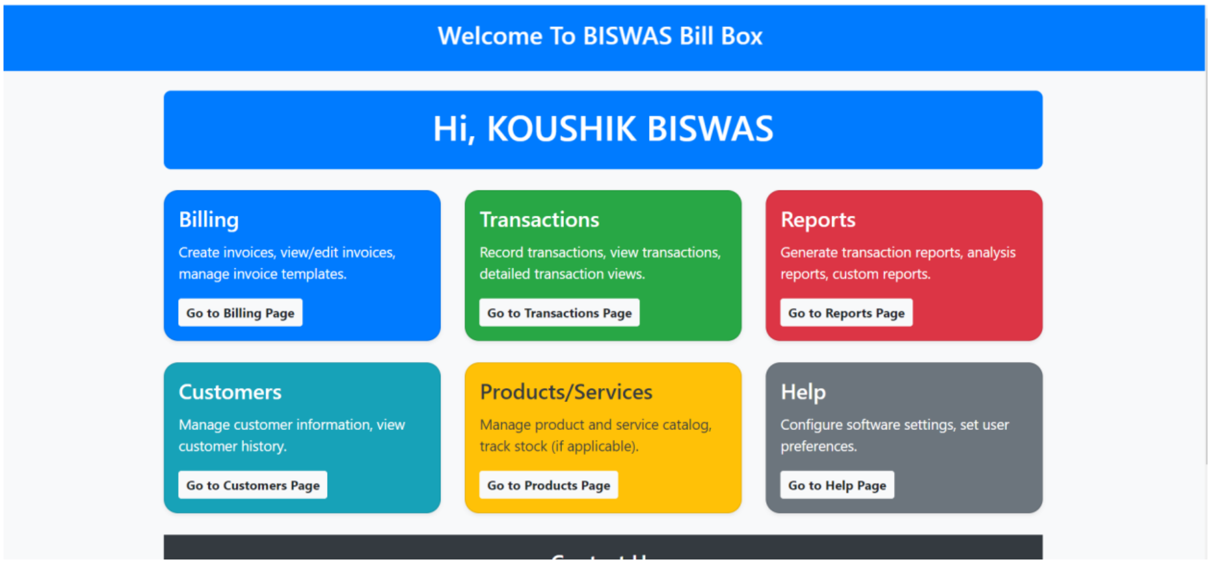
Enter your mobile number in the provided box. For a demo, you can use the number “123456789” without registration.

Step 3: Fill Out the Registration Form

1. **Enter your number**: This will serve as your ID.
2. **Enter your name, company name, address, and pincode**: This information will be used in your invoices.
3. **Click on "Submit & Log in"**.

Step 4: Log In to Your Account

After submitting the registration form, you will be redirected to the login page. Enter the number you used for registration and click on the "Let's Go" button. This will open your dashboard.



Welcome To BISWAS Bill Box

Hi, KOUSHIK BISWAS

Billing
Create invoices, view/edit invoices, manage invoice templates.
Go to Billing Page

Transactions
Record transactions, view transactions, detailed transaction views.
Go to Transactions Page

Reports
Generate transaction reports, analysis reports, custom reports.
Go to Reports Page

Customers
Manage customer information, view customer history.
Go to Customers Page

Products/Services
Manage product and service catalog, track stock (if applicable).
Go to Products Page

Help
Configure software settings, set user preferences.
Go to Help Page

Dashboard Overview

Your dashboard includes the following sections:

1. **Billing**:
    - Generate bills quickly.
    - Add products or services before generating bills.
2. **Transaction**:
    - View recent selling product details.
3. **Reports**:
    - View important data using plots.
    - Reports update instantly after selling or adding a product.
4. **Customer**:
    - View customer details such as phone number, billing date, and bill amount.

5. **Product/Service**:

   - Add products to the system by providing the selling price and buying price.
   - Calculate profit margins instantly after selling a product.

6. **Help**:

   - If you need modifications to your dashboard, click on the Help option and email us.

Conclusion:-

- This documentation provides a detailed overview of each endpoint in the BISWAS Bill Box application, including their URLs, associated views, and descriptions. Use this guide to understand the functionalities and navigate through the application effectively.
- This documentation provides a detailed description of each model used in the BISWAS Bill Box application. These models are the backbone of the application, representing users, products, bills, and invoice product details. Ensure that each model is correctly defined and understood to maintain the integrity and functionality of the application.
- By following this documentation, you should be able to set up, run, and use the BISWAS Bill Box application effectively. Ensure that your directory structure is correct, adjust the Python path as necessary, and run scripts from the project root to maintain the package integrity. Use the provided endpoints, models, and front-end technologies like JavaScript and Bootstrap to create a robust and user-friendly billing system.