BANK MANAGEMENT SYSTEM

Index

| Section | Topic |
| --- | --- |
| **Chapter 1** | **Introduction** |
| 1.1 | Abstract |
| 1.2 | Problem Statement |
| 1.3 | Objectives |
| 1.4 | Scope of the Project |
| 1.5 | Requirements |
| **Chapter 2** | **System Analysis and Design** |
| 2.1 | Literature Survey / Existing System |
| 2.2 | Proposed System (Architecture Diagram) |
| 2.3 | Database Design |
| **Chapter 3** | **Implementation** |
| 3.1 | Project Modules and Functionality |
| 3.1.1 | Homepage |
| 3.1.2 | Login & Registration |
| 3.1.3 | Contact Us |
| 3.1.4 | Dashboard |
| 3.1.5 | Deposit |
| 3.1.6 | Withdraw |
| 3.1.7 | Statement & Reports |
| 3.2 | Brief Code Snippets and Description |
| 3.3 | Security Features |
| **Chapter 4** | **Testing and Results** |
| 4.1 | Implementation & Screenshots (Appendix A) |
| 4.2 | Testing |
| 4.3 | Results and Discussion |
| **Chapter 5** | **Conclusion and Future Scope** |
| 5.1 | Conclusion |
| 5.2 | Future Scope |
| Appendix | Screenshots and References |
| A. | Screenshots |

**Chapter 1: Introduction**

**1.1 Abstract**

This project introduces an **Online Banking Web Application** leveraging Django (Python), HTML/CSS, and supporting technologies. The system was developed to deliver modern banking features such as user authentication, deposits, withdrawals, and account statements. The application emphasizes responsive design, security, and user experience, demonstrating best practices in web programming and database management.

**1.2 Problem Statement**

Most traditional banking processes are time-consuming, paper-based, and limited to physical locations. This project addresses the need for secure, accessible, and user-friendly online banking tools that empower customers to manage finances remotely and efficiently.

**1.3 Objectives**

* Deliver **secure user registration and login**.
* Implement **core banking operations** (deposit, withdrawal).
* Present transaction history with filtering and **export options** (CSV).
* Ensure **mobile and desktop user experience** (responsive design).
* Facilitate administrative oversight through a Django admin interface.

**1.4 Scope of the Project**

The project primarily covers User account management, Transaction processing, and Reporting/statements. Future work can expand into loan management, bill payments, and integrated third-party services.

**1.5 Requirements**

| Type | Detail |
| --- | --- |
| **Software** | Python (Django framework),  SQLite/compatible database,  HTML5, CSS3, Bootstrap |
| **Hardware** | Standard developer workstation |

**Chapter 2: System Analysis and Design**

**Project Structure**

* **Project Name:** BANKMANAGEMENTPROJECT
* **App Name:** WELCOMEPAGE
* **Folders:**
  + templates/ (for base.html, login.html, welcome.html, register.html, contactus.html, success.html)
  + static/images/ (add logo.png, hero.png)
* **Key Django files:** settings.py, urls.py, views.py.​

**2.2 Proposed System (Architecture Diagram)**

Our system strengthens security, simplifies the User Experience (UX), and provides modular, maintainable code. Key features include custom user profiles, transaction audit trails, and real-time feedback using Django messages. The application follows the Django **Model-View-Template (MVT)** architecture.

**2.3 Database Design**

The system relies on four primary models:

| Model | Purpose | Key Fields | Source |
| --- | --- | --- | --- |
| **User** | Django’s built-in authentication | Username (Email), Password |  |
| **UserProfile** | Extends User |  | user (OneToOne) , balance (DecimalField, default 0.00) , accountnumber (Unique, 12-digit) |  |
| **Transaction** | Audit Log |  | UserProfile (ForeignKey) , Type (deposit/withdraw) , Amount , Resulting balance , Timestamp |  |
| **Contact** | User Inquiries |  | name , email , message , is\_resolved (default False) |  |

**Chapter 3: Implementation**

**3.0 Implementation**

1. **Setup Project:**
   * django-admin startproject BANKMANAGEMENTPROJECT
   * cd BANKMANAGEMENTPROJECT
   * python manage.py startapp WELCOMEPAGE
2. **Configure Templates and Static Files:**
   * Project-level and app-level templates folders.
   * Add static/images (logo, hero).
   * Update settings.py for templates, static paths, and include the app.
3. **URL Configuration:**
   * Project urls.py: route admin, home, and include app urls.
   * App urls.py: define paths for login, register, welcome, contact us, success.
4. **View Functions:**
   * Implement views: home, Mylogin, welcome, register, contactus, success.
   * Each view renders its respective template.​
5. **Template Design:**
   * Create base.html for the main layout using sample HTML/CSS from the notes.
   * Use blocks for title, header, and navigation links (login, register, contact us).
   * Create page-specific templates extending base.html.
6. **Styling and Responsiveness:**
   * Use provided CSS rules for navigation, hero section, buttons, etc.
   * Ensure responsiveness using media queries as given in the notes.

**Features**

* Responsive header with navigation (login, register, contact us)
* Main homepage (hero section, call-to-action)
* User authentication pages (login, register)
* Contact and welcome pages
* Illustrations and custom branding with static assets​

**3.1 Project Modules and Functionality**

| **Module** | **Description** | **Key Features** |
| --- | --- | --- |
| **Homepage** | Main interface for user navigation. | Responsive header with navigation (login, register, contact us) and a main hero section. |
| **Login & Registration** | User authentication and account creation. | Auto-generated unique 12-digit account number ; confirmation email notification. |
| **Contact Us** | User inquiry form. | Saves inquiries to the **Contact** model for admin review ; redirects to a success page. |
| **Dashboard** | User landing page after login. | Personalized welcome message with user's name ; navigation to Deposit, Withdraw, and Statement modules. |
| **Deposit** | Adds funds to the account. |  | **Atomic update** of balance ; records transaction. |
| **Withdraw** | Removes funds from the account. | Validates against current balance; **atomic update**; error message for insufficient funds. |  |
| **Statement & Reports** | View transaction history. | Lists all transactions ; filters by month/year/date range; **CSV export**. |  |

**3.2 Brief Code Snippets and Description**

| Module | Code Snippet (Conceptual) | Description | Source |
| --- | --- | --- | --- |
| **Atomic Transaction** | from django.db import transaction with transaction.atomic(): profile.balance += amount profile.save() | Ensures that both the balance update and the transaction record creation are treated as a single, consistent database operation to maintain financial integrity. |  |
| **Account Number Generation** | def generate\_unique\_number(): while True: number = ''.join(secrets.choice(string.digits) for \_ in range(12)) if not UserProfile.objects.filter(accountnumber=number).exists(): return number | Logic to generate a unique 12-digit account number using the secrets and string modules, iteratively checking the database for uniqueness. |  |
| **Email Notification** | from django.core.mail import EmailMultiAlternatives msg = EmailMultiAlternatives(...) msg.attach\_alternative(html\_content, "text/html") msg.send() | Code integration for sending a multi-part email (plain text and HTML) to confirm successful registration and provide account details. |  |

**3.3 Security Features**

The system employs robust security measures:

* Uses **Django's built-in User authentication system**.
* Utilizes **password hashing** and session management.
* Employs **CSRF protection** on all forms (login, register, transactions).
* Stores sensitive SMTP settings securely in a **.env file** and uses python-decouple.
* All core banking modules use the @login\_required decorator.

**Chapter 4: Testing and Results**

**4.1 Implementation & Screenshots (Appendix A)**

The project setup involved starting the project (BANKMANAGEMENTPROJECT) and an application (WELCOMEPAGE). Key folders for templates and static assets were configured.

| Placeholder | Description |
| --- | --- |
|  | Shows the main base.html layout with the hero section and navigation links. |
|  | Displays the styled, center-aligned register.html form with all required fields. |
|  | Shows the content of the automatic confirmation email with the user's name and new Account Number. |
|  | The welcome.html page after login, showing the user greeting and links to the core banking modules. |
|  | The statement.html page, showing the table of transactions, filtering options, and the "Export Statement (CSV)" button. |

**4.2 Testing**

Testing covered unit and manual test cases for registration, transaction limits, form validation, and error handling. The testing phase included running migrations, creating a superuser, and verifying entries in the Django admin.

**4.3 Results and Discussion**

The project successfully implemented a **secure** and **responsive** platform. The use of **atomic transactions** ensures data integrity for core financial operations. The modular Django structure allows for easy maintenance and future expansion.

**Chapter 5: Conclusion and Future Scope**

**5.1 Conclusion**

The project successfully created a scalable, secure online banking platform, demonstrating a strong foundation in core banking features (deposit, withdraw, statement) built upon the reliable Django framework.

**5.2 Future Scope**

Future enhancements include:

* Integration with live banking APIs.
* Addition of loan/payment/credit card modules.
* Implementation of extensive reporting and data analytics.

**Appendix**

**A. References**

* Django Official Documentation
* Python.org
* Project-related academic papers, URLs, and books reviewed