**PROJECT PART 1**

**Names of Students:**

Nikhil Shetty 1001561794

Biswa Nanda 1001558251

Kavya Kesavan 1001495334

Nihar Suryavanshi 1001654583

Sanjana Kudige Anish Kumar 1001526215

Sonali Gajendragadkar 1001518183

Yogesh Maurya 1001520339

Yukti Sharma 1001559939

Thanmai Peddireddy 1001662111

**SECURITY SECTION:**

**Security Requirements:**

List of vulnerabilities in our system:

1. Input Validation
2. Broken Authentication
3. Sensitive Data Exposure
4. Injection
5. Cross-site Scripting

The attacks which could happen by exploiting these vulnerabilities include:

1. Input Validation:
   1. Username Enumeration
   2. SQL Injection
2. Broken Authentication:
   1. Session hijack
   2. Cookie hijack
3. Sensitive Data Exposure:
   1. Weak/No Encryption
   2. SQL Attack
4. Injection
   1. SQL Injection
   2. iFrame Injection
5. Cross-site Scripting

Counter Measure:

1 a. Display appropriate error message to avoid disclosure of usernames.

b. Input sanitization: Implement whitelisting approach. Use of safe API’s and parametrized queries.

2. a. Session isolation and Idle session timeouts

b. Using secured cookies and multifactor authentication

3. a. Encryption of all data in transit and at rest using secure protocols and algorithms.

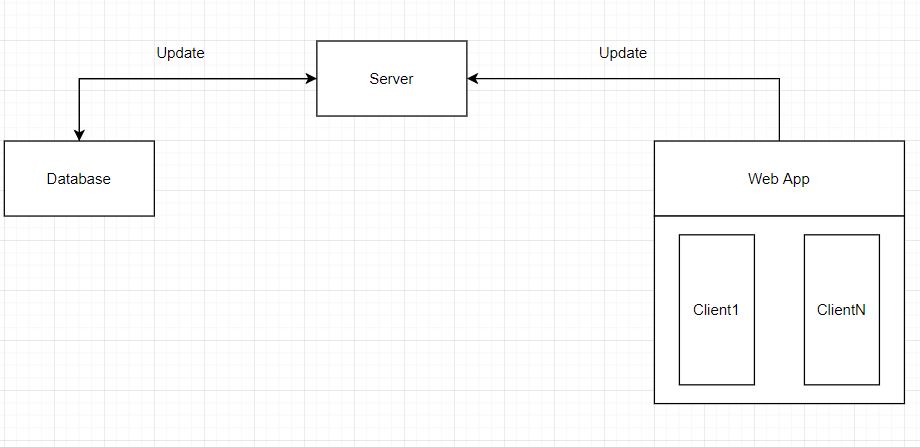
b. User awareness about possible site phishing and iframe injection.

4. Encoding output and skipping untrusted characters. Enabling Content-Security-policy (CSP)

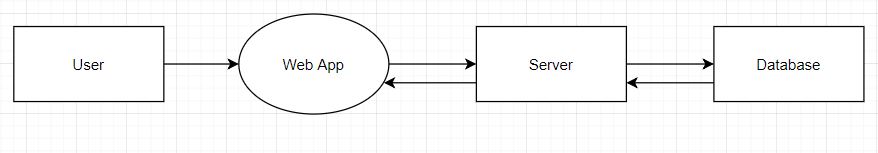
Change History:

|  |  |  |
| --- | --- | --- |
| **Name** | **Date** | **Description** |
| Nihar Suryawanshi | 11/8/2018 | Removed one point from SCRM people table |
| Nihar Suryawanshi | 11/10/2018 | Updated SCRM Technology table |
| Biswa Nanda | 11/16/2018 | Update in testing tool |
| Nikhil Shetty | 11/16/2018 | Update in White space coding standard |

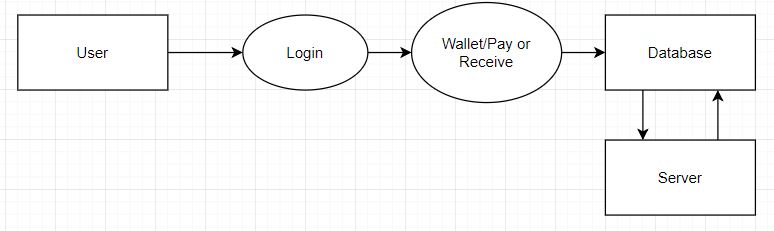
**Security Architecture:**

****

**DFD:**

****

DFD Level 0

****

DFD Level 1

**Threat Modeling:**

Threat modelling and attack surface measurement is an indicator of the system’s security. It is the set of ways in which an adversary can enter the system and potentially cause damage. This in any way does not represent code quality. Thus, the application vulnerabilities can be discovered and exploited.

In our ThomasPay system an attacker may have many ways to attack the system in order to gain access to the system and manipulate it or the data for his/her benefit.

In the previous sections we have described a list of vulnerabilities our ThomasPay system will face.

Following are a few threats posed to the system with a common goal.

1. Phishing:

Attacker may spread a false link/url of the online system to the customers (through emails).

Goal is to capture login information once the user will log-in through the duplicate link. Allowing the attacker to make transactions for their own benefit.

We may be able to avoid this by hosting the system on a secure server (https://) and provide regular awareness to the customers for such attacks.

2. Data Interception or Access:

By means of injections and lack of authentication checks over the database, an attacker can gain access to the database.

Goal is where data can be intercepted or accessed by the attacker. He/she may use database information and perform unauthorized actions and transactions.

We may avoid this by having secured database, restricting the number of people with admin access and having authentication check before database access.

3. Data Theft:

Attacker may keep copy of the essential data in its network which is way outside the ThomasPay network. Attacker’s goal is that data can be sent out of the system and the network with ease for attackers use. He/she with complete information at their disposal may perform unauthorized transactions.

4. Session High Jacking:

Session hijacking is the exploitation of a web session to gain unauthorized access to information and the services of a system.

The attacker takes advantage of an active session and use a sniffer to capture a Session ID. Further, use it to gain unauthorized access to the server.

We can avoid this by using SSL/TLS (Transport Layer Security Protocol) to encrypt data traffic for a particular session. This will avoid sniffing-style attacks. We can have forced session time-outs for sessions which have been idle for a certain amount of time.

5. Cross-Site Scripting:

Cross-site scripting is a way of capturing an active valid Session ID where the attacker can trick client system into running malicious code or a script which may appear to be trustworthy to the client. By this the attacker will gain access to user cookies.

In order to avoid this we will follow and implement XSS prevention rules.

This ThomasPay system aims at being an online portal that allows secure online transactions for purchase and between customers. An attacker’s main aim would be to somehow gain access to the system and further access to the data in order to perform unauthorized transactions for his/her own benefit without raising any flags.

**Supply Chain Risk Management:**

We have categorized three entities where supply chain risk management principals can be applied to make our application more robust to security threats. The possible risk associated with each category are listed with corresponding mitigations.

The Three categories are:

1. People
2. Technologies
3. Miscellaneous

Miscellaneous

People

Technology

People:

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Inadequate Security Knowledge to engineering team. | 1) Trainings to the Engineering team can be given from security experts.  2) Security certifications can be provided to the team members. |
| Trading confidential transactional data | 1) limited access rights to database and system should be provided or as per the need upon manager’s approval.  2) Email monitoring software should be in place. Ex. Teramind.  3) External storages will be prohibited.  4) USB ports will be blocked. |
| ~~Access to Datacenter and Servers~~ | ~~1) No access to physical hardware. Special security clearance required to reach data server.~~ |
| Application user access to data | 1)Enforce minimum privileges to user for accessing internal data used for verifying card holders data and authorize payment based on business rules. |

Technologies:

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Cross site scripting in Python Flask | 1)Use White listing Input validation |
| SQL injection in Python Flask | 1)Use of parameterized query.  2)Use White listing and regular expression for input validation. |
| Host Header Forgery in python Flask | 1)Use post request.  2)Use unique token to validate the user and requests |
| CrossSite Request forgery Flask | 1) Use unique token to validate the user and requests |
| Risk in session hijacking | 1)Use session timeout after period of inactivity |
| Possible DOS attack | 1)Provide anti DOS mechanism to prevent and detect DOS attack. |
| Restricting URL | 1)Do not use default URL’s  2)Use whitelisting in redirecting URL from servers |

Miscellaneous

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| 3rd party data interaction risk.  Ex: payment gateways, banks. | 1)data should be restricted based on the business rules.  2)security guidelines and requirement will be documented in the contract. |
| Accidental information leakage.  Poor exception handling | Security module should be build up on the top f the standard trace to limit the information. |
| Use of public hotspot or internet for transfer services | Educate users to not use public internet and hotspot for the application. |
| Card holders data security | Enforce strong multi-factor authentication for access to critical systems with credit/debit card holders data. |
| Transactional errors caused by system | 1)Strong logging system should be used so as to restore and rectify transactional glitches. |
| Weak Pin Exposing to brut force attack | 1)Restrict password with various policies to generate strong password. |
| Compromised Payment gateways | 1)Enforce point to point Encrypted communication between host and gateway. |
| Weak server authentication between internal systems calls | 1)Special authentication mechanism should be designed inorder to safeguard this problem. |

Sort term and Long term SCRM:

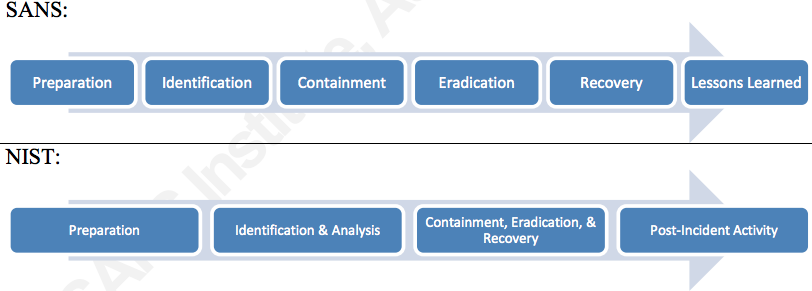
Further we have added few categories which should be handled separately based on the time frame.

Short Term:

|  |  |
| --- | --- |
| Risk | Mitigation |
| Server Failures | Backup servers should be ready at all point of time. |
| Business continuity in case of Natural Calamities and Disasters | 1)Provisions and training to employees to work from multiple office location across country.  2)Training to multiple teams to handel critical job roles. |

Long Term:

|  |  |
| --- | --- |
| Risk | Mitigation |
| Lot of transactional data | 1)Transaction data will be stored for ten years. After that the data would be compressed and stored in safe data vault.  2)Data can be used for Analysis and trend modelling purpose strictly on Business decision |
| User Demographic data | 1)User demographic data will be deleted upon 5 years of inactivity period.  2)user needs to make new profile with Thomas Pay. |
| Zero Day challenges | Use SANS and NIST methodologies for zero day vulnerabilities. |

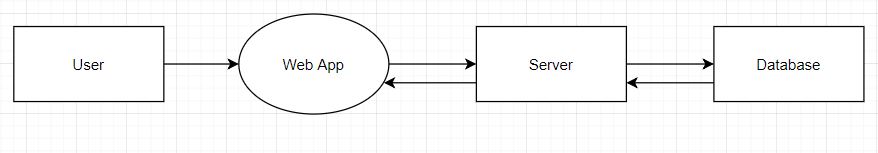


Refernce: https://www.sans.org/reading-room/whitepapers/incident/responding-zero-day-threats-33709

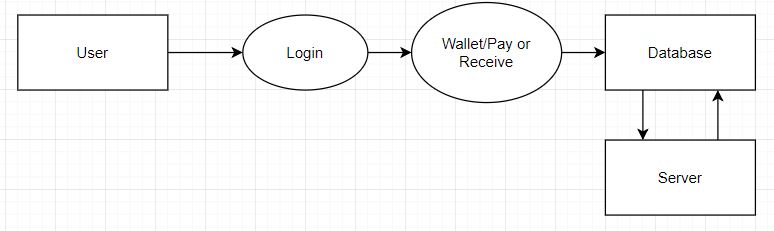
**Analysis and Design**

Informational/Functional/Behavioral Modeling:

Data Flow Diagrams



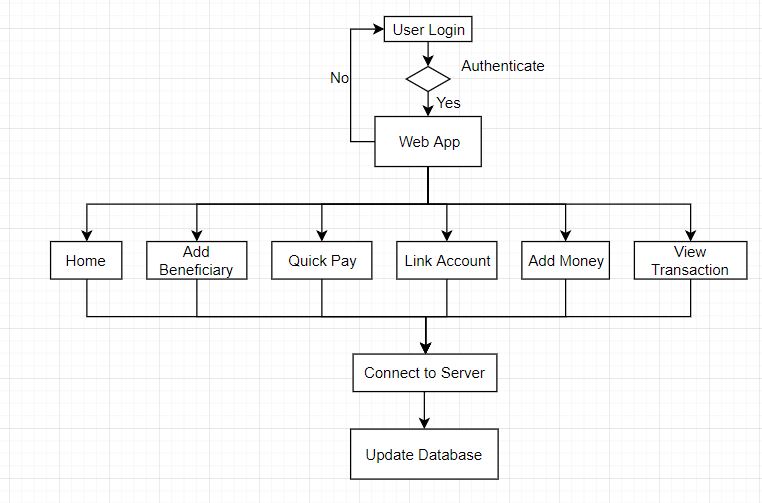
DFD Level 0



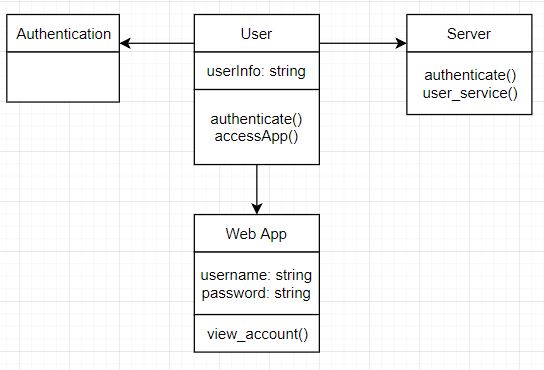
DFD Level 1

Unified Modeling Diagram:

Activity Diagram:



Class Diagram:

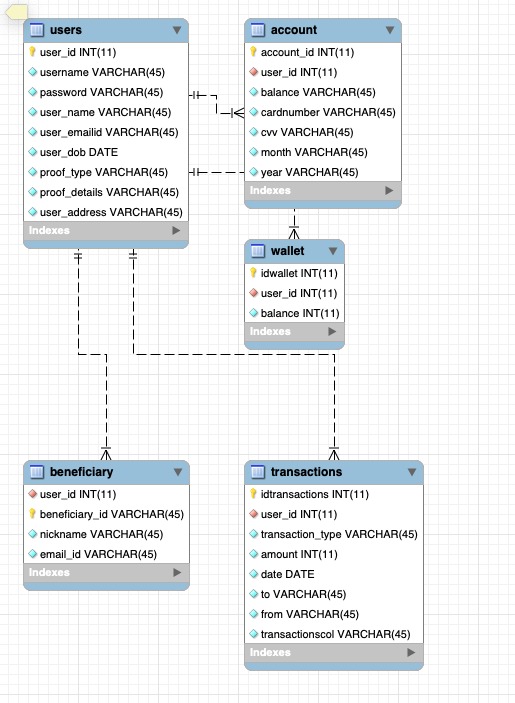


Design Details Rationale:

Our system has following components:

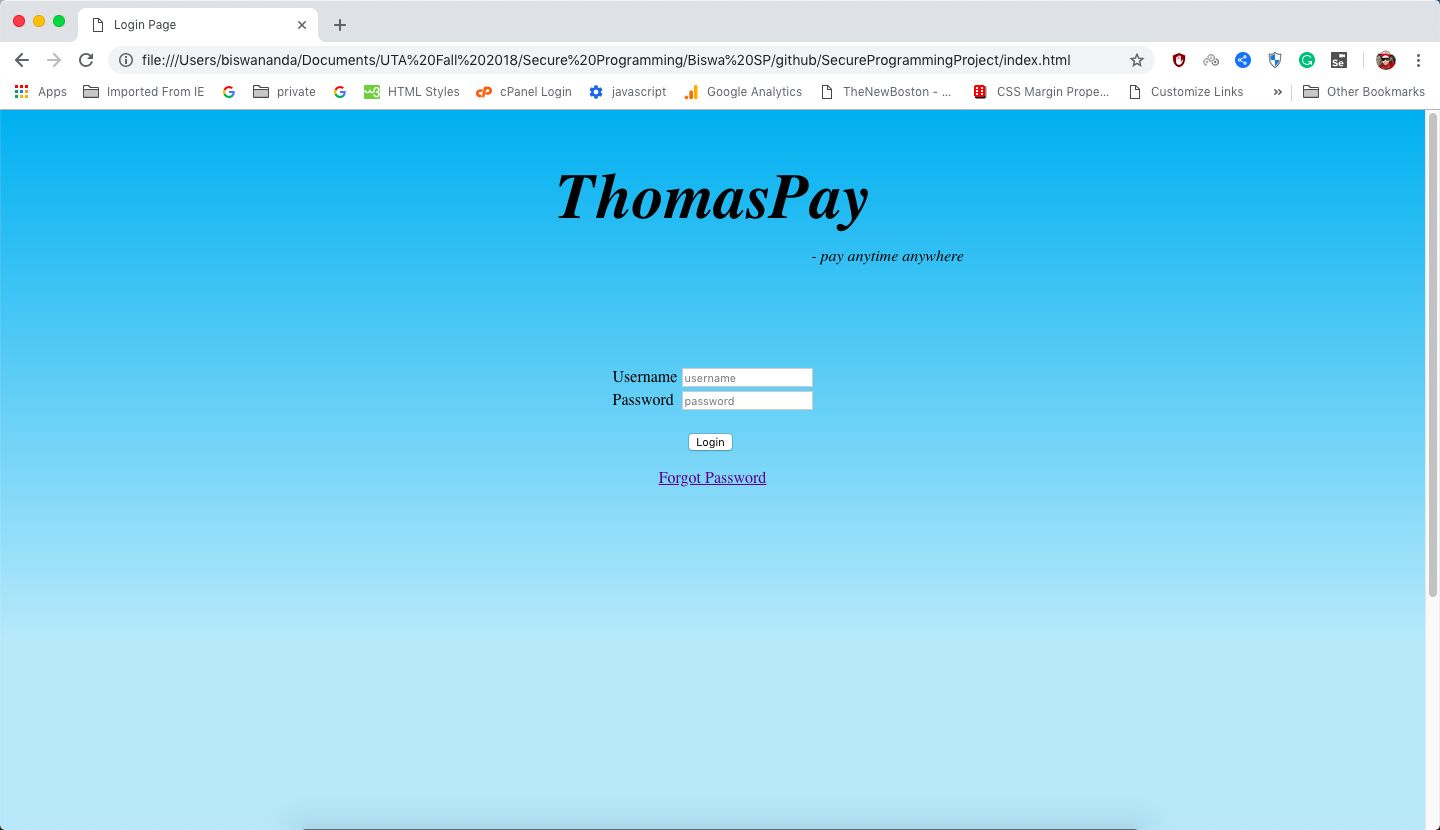
1. **Web Application:** We are making a pay pal like web application, to send and receive the money, we are adding other functionalities like view transactions, wallet management, adding beneficiaries, linking bank account etc.
2. **Database:** We will have pre-populated database of users and their passwords, our database will also have details about user’s transactions, wallet, linked bank accounts.
3. **Server:** We will also have a backend server to host our web application as well as to facilitate the working of our system efficiently.

Database Design

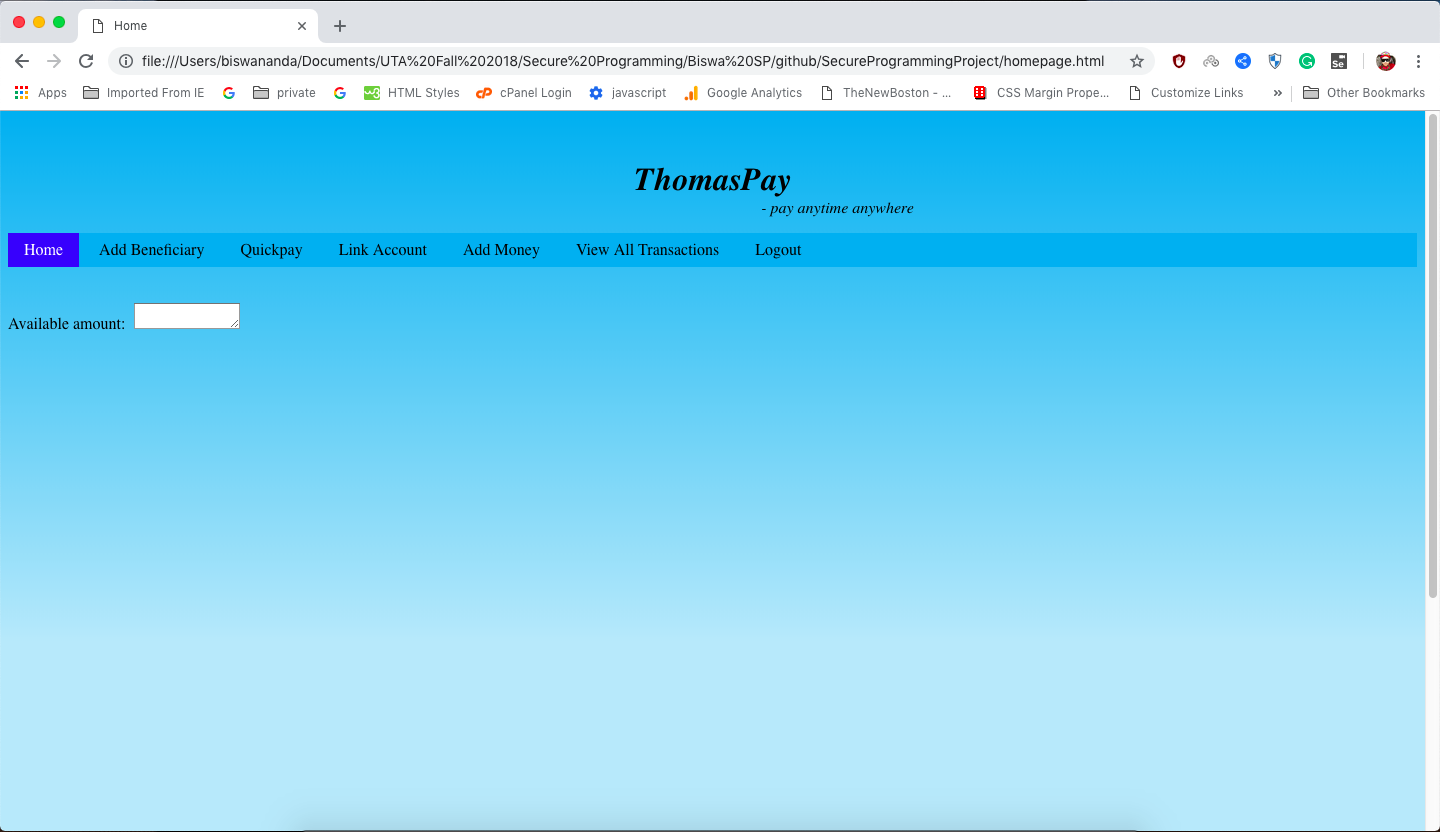


UI Mockups:

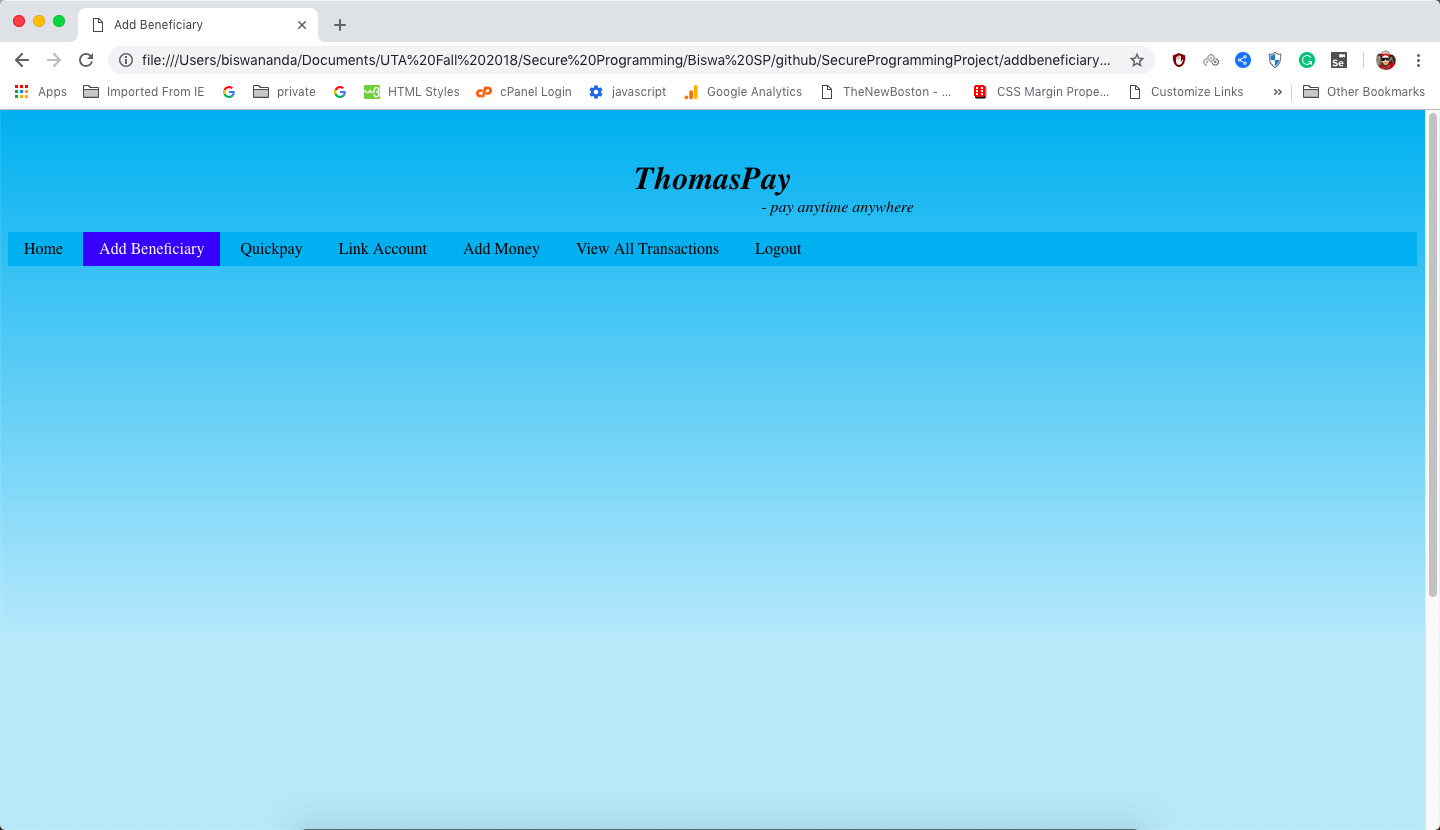
Login/Index Page



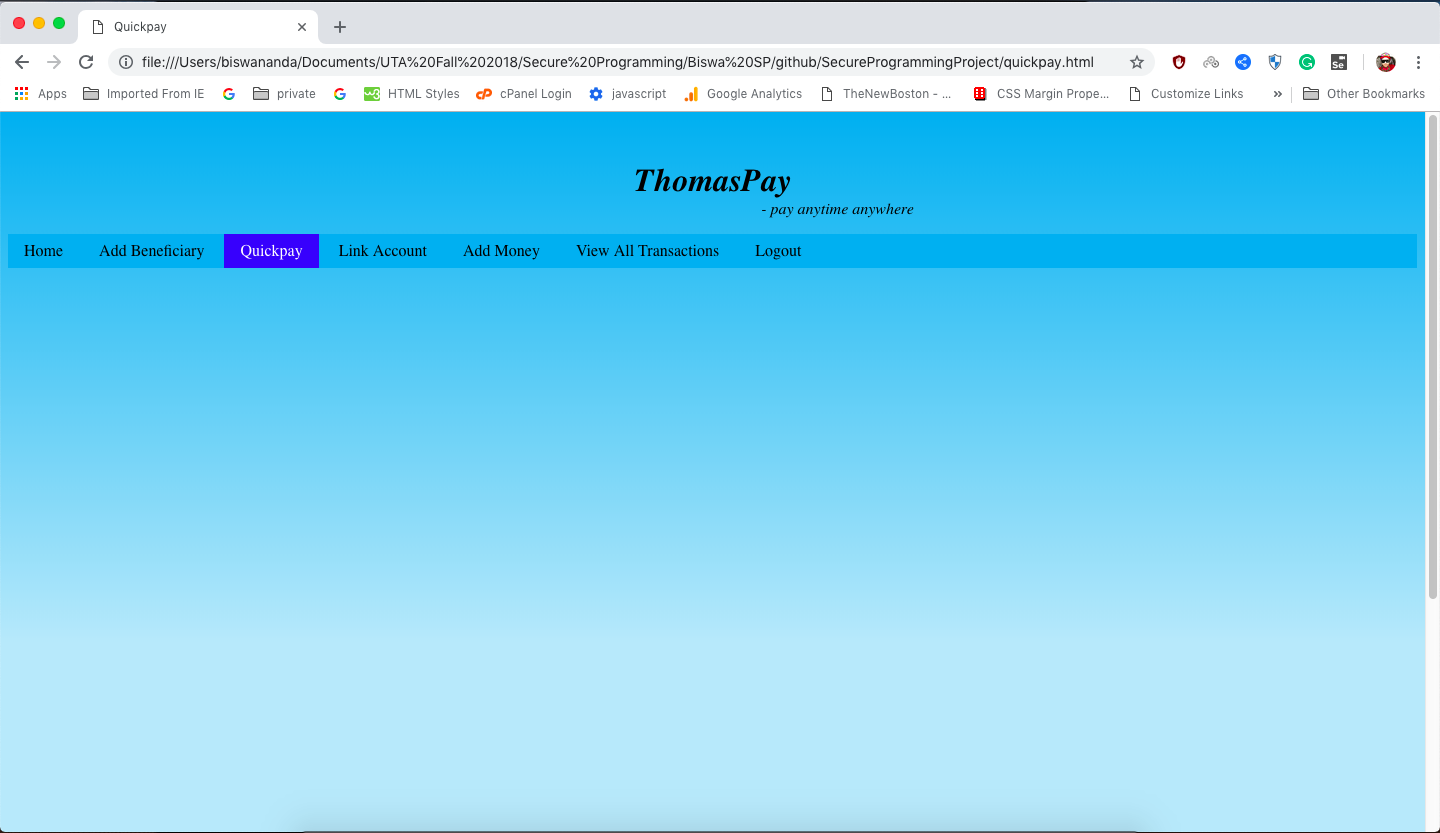
Home Page



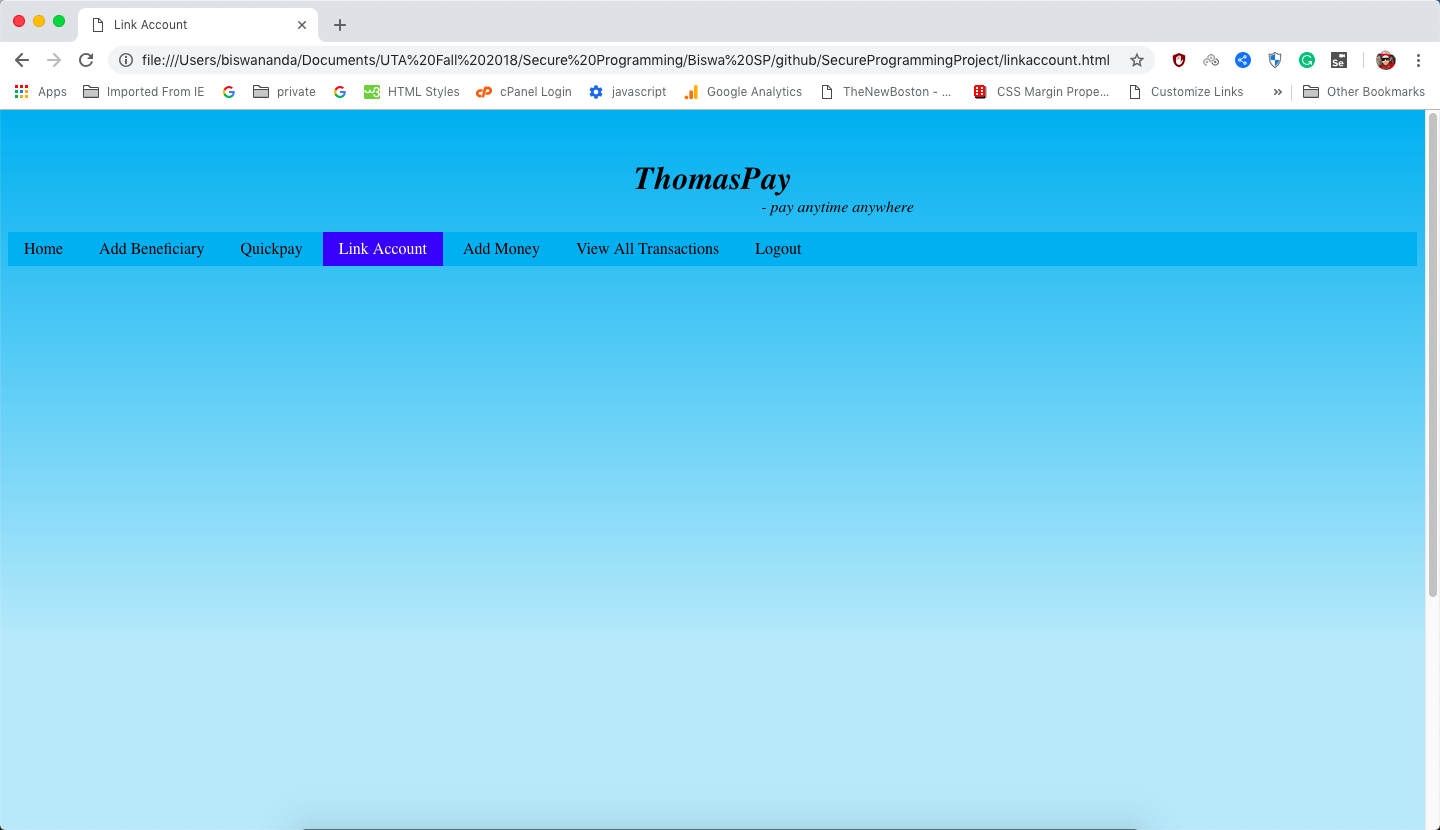
Add Beneficiary Page



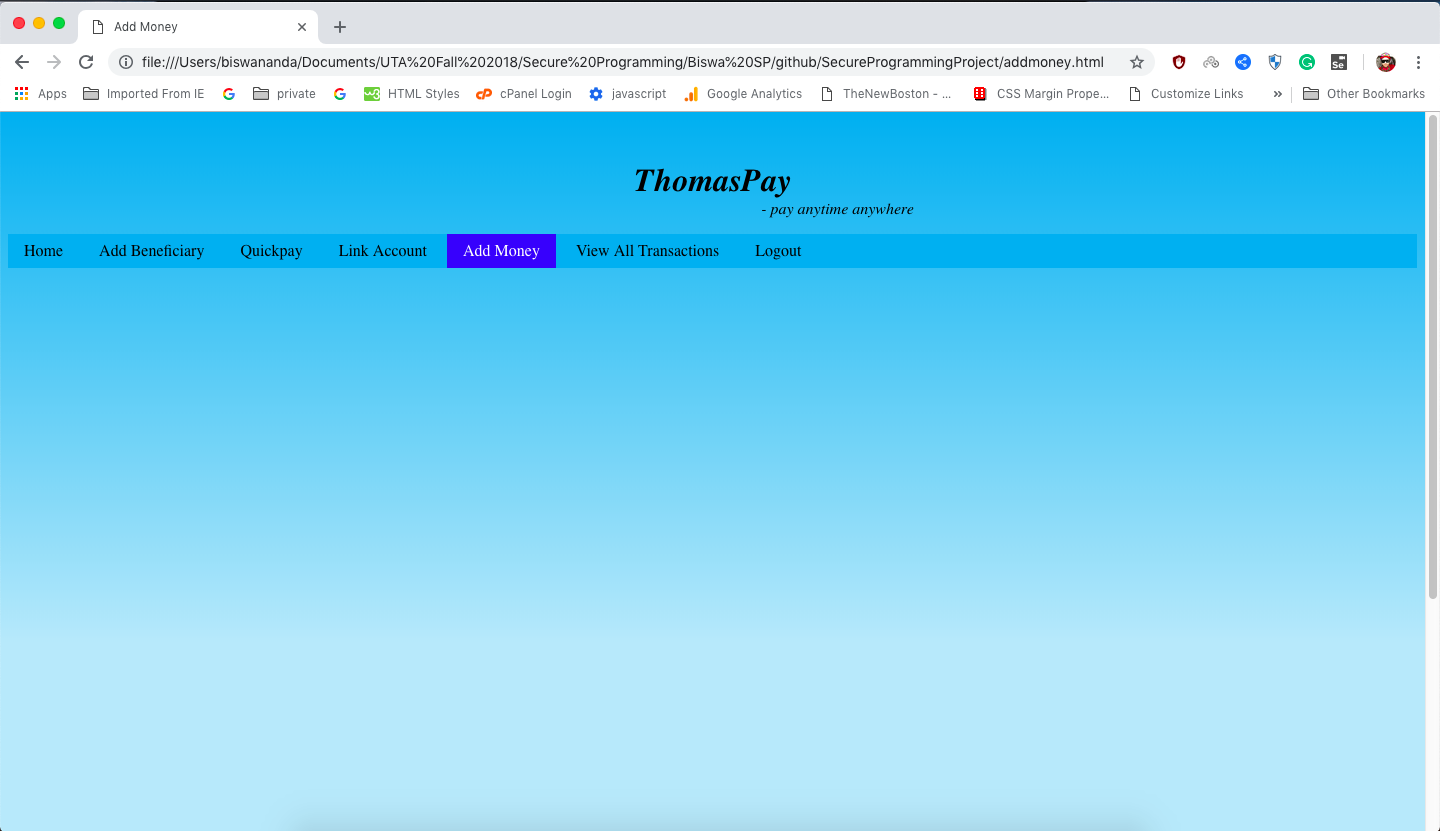
QuickPay Page



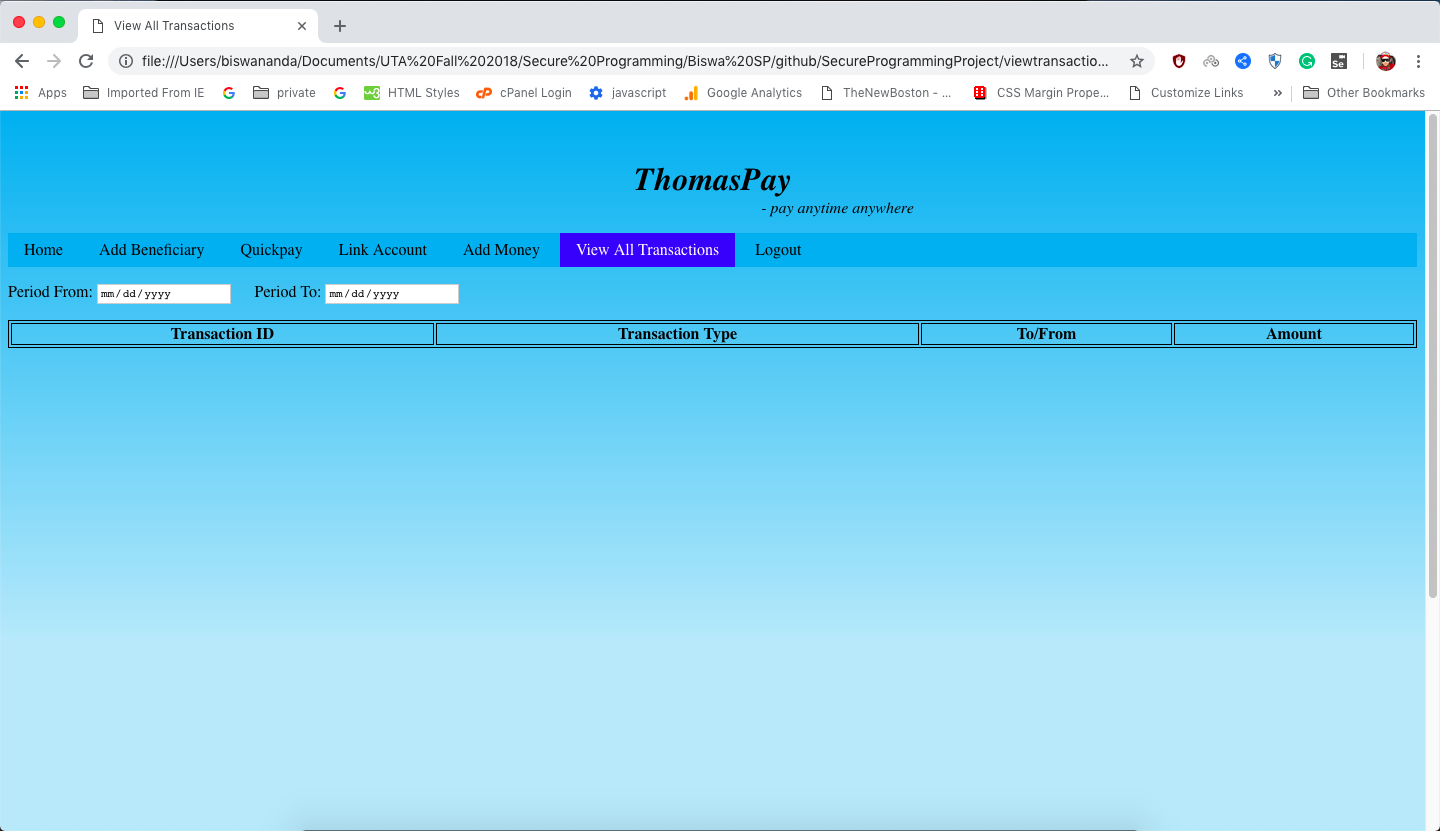
Link Account



Add Money



View All Transactions Page



**Development Strategy:**

**Configuration Management Plan**

1. **Roles and responsibilities for change control boards**

The primary purpose of Configuration Control Board (CCB) is to review and approve proposed changes to the functionality or architecture by involving the entire group. The CCB shall consist of all the members as we don’t have any hierarchy among the members of the group.

During the weekly meeting we discuss about the changes required and vote for it. The general agenda of the meeting will include status reports from development efforts, change requests, and problems faced while development.

The CCB is responsible for review of the problem and request for change that come through. The CCB will determine the changes needed and a general timeline for development and deployment, which initially will be based upon estimated time of development. The CCB will coordinate the deployment schedule for new version of the software.

1. **Identification scheme for configuration items**

**Module Identification:**

All modules in our system will be named in accordance with WHC Software Practices. The naming convention is as follows:

XXYZZZZ.999

where XX - identifies the name of the module

Y - identifies the module as

M - for a menu identifier

D - for a display identifier

U - for an update identifier

R - for a report identifier

ZZZZ - sequential number for unique identification

1. **Process and tools**

We are use git for the change management of this project. Each member of the group has their own repository which is create as a branch of the master repository of the project. All the changes should be added to their own branch before merging the code to the master.

For each of change management we are using git desktop. The main thing to keep in mind while start a new development is to get the latest version of the code from the master.

1. **Baseline management**

The objective of establishing a baseline is to define a basis for further system life cycle process activity and allow reference to, control of, and traceability among configuration items and to requirements. It serves as the common reference that all system development activity is built on and dictates to the development team the changes that are to be implemented.

* Baselines shall be established for the configuration items. Developmental baselines will be established to aid in controlling the software development life cycle processes.
* The changes to the Production baseline require review and approval by the CCB.

Baselines are established in a system development effort to define a formal departure point for controlling future changes that affect performance or design. Product baselines should be reviewed and approved with an approval memo and attachments for the description of any discrepancies that are part of the release.

The following items should go in baseline:

* A secure login for valid registered user.
* Successfully add beneficiary to your account which is another valid user.
* Link your credit card.
* Send and receive money successfully.
* Add money to your wallet.

**Coding Standards:**

The below coding standards have been enhanced to match PEP8 coding style.

PEP8 (Python Enhancement Proposals) is a coding style used in python projects for better code understanding and readability and to write the code in pythonic way.

**Reference:** https://www.python.org/dev/peps/pep-0008/#designing-for-inheritance

[Indentation](https://www.python.org/dev/peps/pep-0008/#id17):

Use 4 spaces tab per indentation level.

[Maximum Line Length](https://www.python.org/dev/peps/pep-0008/#id19)

Limit all lines to a maximum of 79 characters.

For flowing long blocks of text with fewer structural restrictions (docstrings or comments), the line length should be limited to 72 characters.

[Should a Line Break Before or After a Binary Operator?](https://www.python.org/dev/peps/pep-0008/#id20)

Use the line break before the operator.

Ex:

income = (gross\_wages

+ taxable\_interest

+ (dividends - qualified\_dividends)

- ira\_deduction

- student\_loan\_interest)

[Blank Lines](https://www.python.org/dev/peps/pep-0008/#id21)

Surround top-level function and class definitions with two blank lines.

Method definitions inside a class are surrounded by a single blank line.

Extra blank lines may be used (sparingly) to separate groups of related functions. Blank lines may be omitted between a bunch of related one-liners (e.g. a set of dummy implementations).

Use blank lines in functions, sparingly, to indicate logical sections.

[Source File Encoding](https://www.python.org/dev/peps/pep-0008/#id22)

Code in the core Python distribution should always use UTF-8

Files using ASCII (in Python 2) or UTF-8 (in Python 3) should not have an encoding declaration.

[Imports](https://www.python.org/dev/peps/pep-0008/#id23)

Import should be on separate line.

[Module Level Dunder Names](https://www.python.org/dev/peps/pep-0008/#id24)

Module level "dunders" (i.e. names with two leading and two trailing underscores) such as \_\_all\_\_, \_\_author\_\_, \_\_version\_\_, etc are allowed.

[String Quotes](https://www.python.org/dev/peps/pep-0008/#id25)

In Python, single-quoted strings and double-quoted strings are the same.

Use Double Quotes strictly.

Ex: “hello world”

Whitespace in Expressions and Statements

* Avoid extraneous whitespace in strings.
* Use single space in if, for and other conditional and iterative statements.
* Use single space in normal statements. ~~Don't use spaces around the = sign~~

Ex: a = a + b

[Comments](https://www.python.org/dev/peps/pep-0008/#id30)

* Use Date and Developer name while adding the comments for any addition or enhancement is done in the code.
* Use block comments whenever necessary.
* Leave one line above and below the comments for better readability.
* Use Inline comments sparingly.

[Naming Conventions](https://www.python.org/dev/peps/pep-0008/#id34)

* **Class name**: Start with Capital non numeric and special character. *Ex: Class\_name.*
* **Method name**: Starts with small letter non numeric and special character. *Ex: method\_name*
* **Functions name**: Start with Capital non numeric and special character*. Ex: Function\_name.*
* **Global variable name**: Start with Capital non numeric and special character. *Ex: Global\_var.*
* **Local variable name**: Starts with small letter non numeric and special character. *Ex: local\_var*
* **File naming:** Starts with small letter non numeric and special character. *Ex: file\_name*
* **Constants**: Should have all capital letters *ex: CONST*
* **Exceptions:** Should start with capatil letter and non numeric and special character. *Ex: Sample\_exception*
* Only special character allowed is underscore“\_”

[Descriptive: Naming Styles](https://www.python.org/dev/peps/pep-0008/#id36)

The following naming styles are commonly distinguished:

* CapitalizedWords (or CapWords, or CamelCase -- so named because of the bumpy look of its letters [[4]](https://www.python.org/dev/peps/pep-0008/#id11)). This is also sometimes known as StudlyCaps.

Note: When using acronyms in CapWords, capitalize all the letters of the acronym. Thus HTTPServerError is better than HttpServerError.

* mixedCase (differs from CapitalizedWords by initial lowercase character!)

[Names to Avoid](https://www.python.org/dev/peps/pep-0008/#id38)

Avoid python keywords.

Ex: ELSE.

**Software Development Plan**

**Software Project Management Plan**

1. **Introduction**

This introduction provides background information for the rest of the document. It briefly describes the project, the client deliverables, the project milestones, and expected document changes.

* 1. **Project overview**

This project is to create a prototype PayPal. The user will have to access the website using a username and password. If the user is visiting the website for the first time, then he/she must create an account or register in the system. The service with sending and receiving money will be available. The system would also provide a Wallet feature where the direct transactions can be done if the payee is already added. The account balance will be available and the pending and posted transactions as well. An exception message shall be generated, and the account will be blocked if the user makes an invalid login for up to three times.

* 1. **Project deliverables**

1. Preliminary Project Plan

2. Requirements Specification

3. Analysis [Object model, Dynamic model, and User interface]

4. Architecture Specification

5. Component/Object Specification

6. Source Code

7. Test Plan

8. Final Product Demo

* 1. **Evolution of this document**

This document will be updated as the project progresses. Updates should be expected in the following sections:

1. ***References*** - updated as necessary
2. ***Definitions, acronyms, and abbreviations*** - updated as necessary
3. ***Organizational Structure*** will be updated as the team leaders are assigned for each phase.
4. ***Technical Process -*** this section will be revised appropriately as the requirements and design decisions become clearer
5. ***Schedule –*** as the project progresses, the schedule will be updated accordingly

* 1. **Definitions, acronyms, and abbreviations**
     1. UML – Unified Modeling Language

1. **Project organization**
   1. **Process model**

The process used for this project will be a Hybrid Waterfall-Yoyo model such that each stage of the waterfall allows us to update the project plan and other deliverables for missing areas or correctness.

We will use UML tools to create the system model and the subsequent breakdown of the design.

* 1. **Organizational structure**

Team Members –

* + 1. Nikhil Shetty
    2. Biswa Nanda
    3. Kavya Kesavan
    4. Nihar Suryavanshi
    5. Sanjana Kudige Anish Kumar
    6. Sonali Gajendragadkar
    7. Yogesh Maurya
    8. Yukti Sharma
    9. Thanmai Peddireddy
  1. **Organizational boundaries and interfaces**

Team leaders during each phase will be responsible for coordinating team meetings, updates, communications, and team deliverables.

* 1. **Project responsibilities**

For primary responsibilities per phase, please refer to section 2.2. Ultimately the entire project team is responsible for the successful delivery of the product.

Team member assignments per deliverable according to expertise.

|  |  |  |
| --- | --- | --- |
| Duration | Name of members | Deliverable Description |
| Week 1 | All team members | Project Plan |
| Week 1 | Sonali, Sanjana | Requirements Specification |
| Week 2 | Yukti, Nikhil | Analysis |
| Week 2 | All team members | Architecture Spec |
| Week 3 | Kavya, Thanmai | Component/Object Specification |
| Week 3 | Yogesh, Nihar | Source Code |
| Week 4 | Nikhil, Biswa | Test Plan |
| Week 4 | All team members | Final Deliverable for part 1 |

1. **Managerial process**
   1. **Monitoring and controlling mechanisms** 
      1. Weekly project status meetings
      2. Shared document repository
      3. Project tracking by MS project plan
      4. Tracking utilizing baselines in MS project

**Software Development Cycle Testing Approach**

**Introduction:**

ThomasPay is the safer, easier way to pay online without revealing your credit card number. Our main aim is to build a system which provides user an easy way to make transaction in a secure fashion

**Test Scope:**

* The main functionality which need to be tested is sending and receiving money in a secure way
* Login functionality with the encrypted password in the database
* Quick pay functionality for money transfer to the beneficiary
* Addition of beneficiary account (verification of user existence using email id)
* Add Money/balance to the account
* For security feature we will check if the data which been stored in the system is stored securely (basically encrypted).
* Penetration testing

**Tactics:(TimeLine for each task)**  
Biswa and Nihar we will be handling Testing the functionality of sending and receiving the money in secure way.

Sonali and Sanajana we will be testing with new contact adding

Yogesh and Thanmai will be handling With bank details and linking them

Yukti, Kavya and Nikhil will test that the data which is been send in an encrypted format.

Yukti, Nikhil and Yogesh will be handling the pen test making sure that the application is not vulnerable to attack.

**For Example**, if you have mentioned that you will be testing the existing interfaces, what would be the procedures that you would follow to notify the key people to represent their respective areas, as well as allotting time in their schedule for assisting you in accomplishing your activity?

**Test scope**

The test scope comprises of the functionalities of the application:

* login functionality with the encrypted password in the database
* Addition of beneficiary account (verification of user existence using email id)
* Quick pay functionality for money transfer to the beneficiary
* Add Money/balance to the account
* View Transaction

**Test Environment**

* Pycharm (Python)
* Web browser (Chrome/Firefox)

**Testing Tools**

* ~~Selenium IDE~~
* ~~Selenium Webdriver~~
* ~~Findbugs~~
* ~~Flaw Finder~~
* Pylint
* Python taint
* bandit

**Release Control**

* Git

**Risk Analysis**

Risk for the application can be explained or analyzed by understand the vulnerability of the code. Understanding the concept of the buffer overflow, we must take care of the overflow vulnerability with proper coding standards and will also make use of the static code analysis techniques to understand the flaws in the program which will make our program free of loopholes that can minimize the penetration or illegal access to the system.

**Overall Review**

For each major group of features or feature combinations, specify the approach which will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools which are used to test the designated groups of features. The approach should be described with sufficient details to permit identification of the major testing tasks and estimation of the time required to do each one. Unit testing will be done component wise, or we can say module wise, we have divided our group and each group will be responsible for respective component. Integration Testing will be performed once the application is ready and all the modules are merged, this will test for all functionality testing of the project.