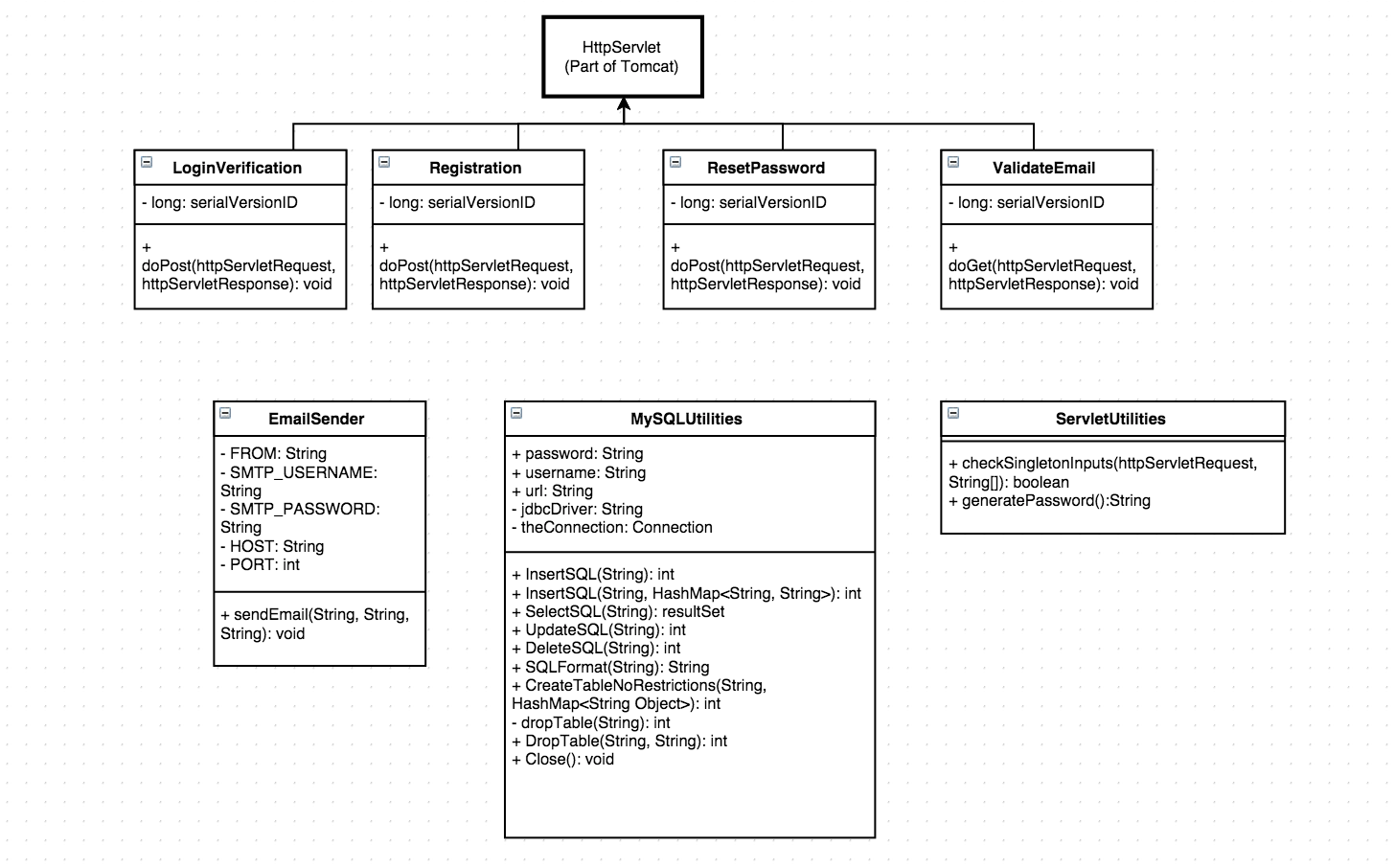
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Iteration | Version | Dates | Comments |
| Mel Chi | 1 | 1.0 | 9/17-9/24 | Added UtilityClass, MySQLUtility Class notes  JUNIT Test |
| Anthony Quigley | 1 | 1.0 | 9/17-9/24 | High Level - Services, Servlets, Database |
| Mitch | 1 | 1.0 | 9/17-9/24 | High Level - All Front-End Components, Diagrams |
| Tim Parzynski | 1 | 1.0 | 9/17-9/24 | JUNIT Test |

• Analysis and Design o Architectural Analysis • High level system components, processes, services, threads, servers etc. o Detailed Design • Static design o Database design o Class diagrams • Dynamic design o Sequence / State diagrams

• Implementation o Code o Comment o Structure and style of code o Unit tests



### High Level

* Services
  + AWS CloudFormation to handle our tech stack
  + AWS RDS MySQL Database
  + AWS EC2 server for managing front end callbacks
    - Running Tomcat to host java servlets
    - Communicates with MySQL database in RDS
  + AWS SES for sending emails on Phoodie’s behalf
* Servlets
  + Login Verification
  + Account Registration
    - Also sends an email with a validation link
  + Reset Password
  + Validate Email
* Front-End Components
  + Implement framework 7, and AJAX in javascript to interact with server
  + Login Page
    - Basic login in that allows user to login, create account, or recover password
  + Registration Page
    - Allows a user to register an account. Asks for Username, password, and email
  + Email Validation Page
    - Simple page that shows user they validated their email
  + Forgot Password Page
    - Allows user to request email to recover password
  + Home Page (with Lorum Ipsum)
    - Basic functionality for each user

Front-End

Login Page

Implement framework 7’s javascript and css to create a login page

### UtilityClass.java

Includes a random password generator, takes any alphanumeric character

### MySQLUtility Class

1. InsertSql(sql string) or (string tablename, hashmap <column name, value>
   1. Can take both a insert sql command or Table Name, and values that are listed in a hashmap
2. SelectSQL(sql string)
   1. Takes the SQL string and returns a result set
3. DeleteSQL(sql string)
   1. Takes the SQL string and executes it
4. CreateTableNoRestrictions(TableName, Hashmap<String, Object>
   1. Creates a generic table
5. DropTable(tablename, password)
   1. Drops the tablename given, password to be added on as well to prevent drops

### Database Design

UserInfo

* Username
  + Primary Key
* Password
* Email
* Unverified Hash

### Junit Tests

### 

|  |  |
| --- | --- |
| Test Case ID: SQL\_1 | Test Title: InsertSQL |
| Description | |
| Tested whether or not we could add new records to a table. | |
| Pass/Fail | Pass |

### 

|  |  |
| --- | --- |
| Test Case ID: SQL\_2 | Test Title: SelectSQL |
| Description | |
| Tested whether or not we could select fields from the database and return a result set. | |
| Pass/Fail | Pass |

### 

|  |  |
| --- | --- |
| Test Case ID: SQL\_3 | Test Title: DeleteSQL |
| Description | |
| Tested whether or not we could delete fields, then check if it was deleted with select. | |
| Pass/Fail | Pass |

### 

|  |  |
| --- | --- |
| Test Case ID: SQL\_4 | Test Title: CreateTableNoRestrictions |
| Description | |
| Tested whether or not we could create a basic table with no advanced options (IS NOT NULL, etc) | |
| Pass/Fail | Pass |

### 

|  |  |
| --- | --- |
| Test Case ID: SQL\_5 | Test Title: DropTable |
| Description | |
| Tested whether or not we could delete tables from the database. | |
| Pass/Fail | Pass |