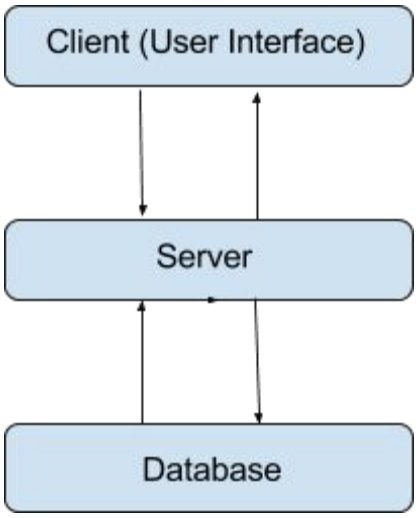
INCREMENTAL AND REGRESSION TESTING

➔ Classification of components



Classification of Components: The overall project uses three major components. A web server that handles interaction of the user to and from the database. Secondly there is an SQL database that will house and store everything needed for a trip to be planned as well as user information stored securely with hash protected passwords. Lastly there is a web based Client interface that the user will interact with, input their preferences and see trips suggested by the site.

- ➔ Client user interface:
  - Input: User inputs data into forms that are used later to help plan trips
  - Output: the data from the user is used to populate tables in the database
  - Dependencies: HTML5, JQuery
  - Dependents: Users
- ➔ Server:
  - Input: User data from the user interface
  - Output: Views for the user interface, updates the database
  - Dependencies: Vert.x3, Openshift
  - Dependents: Client user interface
- ➔ SQL Database:
  - Input: data and sql queries from the server and users
  - Output: responses to the Server from queries
  - Dependencies: MYSQL, JDBC connection
  - Dependents: Server

Form of Incremental Testing

Bottom-up Testing: Considering the structure of our product, we concluded that bottom-up testing, which would involve testing each module and component first and then testing the product on the whole, would be easier for us to debug and build upon. Currently, we have very few interconnected modules and this gives us enough flexibility that enables individual testing. We can test our User Interface, Database and server as individual components first. integrate them with each other and test the application as a whole. For the testing of our UI, we are trying to reload the pages to confirm that there are no crashes in our HTML script and all the UI components load correctly. For the database testing, we are trying to populate our tables with dummy data. We are using JDBC to populate different tables and making sure that the information is getting stored as needed. Also, for our server, we are using dummy data to test the communication between our UI and the database. We are retrieving the data we stored in our database tables and displaying it on our UI, making sure there are no crashes in the process and accurate data is being displayed on the web console.

MODULE	Component A - SQL Database

Incremental Testing			
Defect No.	Description	Severity	How to Correct
1	Application should not crash when trying to retrieve data from empty tables	1	Create an exception to check if the data table is empty.
2	Application should not crash when pushing data into tables which don't exist in the database	1	Create an exception to check if the table exists in the database.
3	The server reads data from the UI and populates the appropriate database tables. Data is being populated into the wrong table or no data is being written to any of the tables.	2	Fix the JDBC connections so that the correct tables are being populated.

Regression Testing

Defect No.	Description	Severity	How to correct
1	Fixing data retrieval from an empty table causes the UI to crash.	1	Catch the exception thrown and display an error message on screen
2	Accessing/writing data from/to tables which don't exist causes the server to go down.	1	Create a new table if exception is thrown and populate it accordingly.
3	Data retrieval from the wrong table causes the User Interface to display wrong data.	2	Make sure the JDBC connection point to the correct tables. Can only be done by inspection of code.

MODULE	Component B - Client Interface

Incremental Testing			
Defect No.	Description	Severity	How to Correct
1	Error pages are not displayed, user gets a page with only error text on it.	2	Implement a failureHandler server side to serve an appropriate page.
2	Users are not returned after form after error, equivalent to application crash	1	Implement server-side local error catching and client side form checking
3	Users are dealt wrong hotels and restaurant choices from their preferences	2	Improve matching algorithms

Regression Testing			
Defect No.	Description	Severity	How to Correct
1	Creating dedicated error page causes it to show when not needed	2	Ensure Error page is only being displayed when there is an error
2	After returned from error form, users errored data are still set and user can't change preferences	2	Users preferences that causes an error are displayed and their preferences are reset for user to change
3	Improved matching algorithms cause crash in connecting to database	2	Make sure algorithm can still connect to database

MODULE	Component C - Web Server

Incremental Testing			
Defect No.	Description	Severity	How to Correct
1	Server stuck on 404 error from hotel API	1	Make server compatible with hotel API
2	Server drops connection from client	1	Use TCP for connecting to clients and ensure stability for web server

Regression Testing			
Defect No.	Description	Severity	How to Correct
1	Fixing 404 error from hotel API causes client to crash	1	Have a readable error from server to client displaying error
2	Using TCP connection makes server stuck in a loop listening to client and cant read from database	1	Use multiple processes to listen to client and connect to database