

IoT Fuzzer

Functional Specifications and Methodology

**Project Advisor:**

Dr. Haroon Mehmood

**Group Members:**

Malik Abdulaziz 17L-4227

Usama Arshad 17L-4244

Abdul Samad 17L-4237

Husnain Maqsood 17L-4228

National University of Computer and Emerging Sciences

Department of Computer Science

Lahore, Pakistan

Table of Contents

[Table of Contents i](#_Toc23336587)

[List of Tables ii](#_Toc23336588)

[List of Figures ii](#_Toc23336589)

[1. Introduction 2](#_Toc23336590)

[1.1 Purpose of this Document 2](#_Toc23336591)

[1.2 Intended Audience 2](#_Toc23336592)

[1.3 Definitions, Acronyms, and Abbreviations 2](#_Toc23336593)

[2. General Description 2](#_Toc23336594)

[2.1 User Characteristics 2](#_Toc23336595)

[2.2 Domain Overview 2](#_Toc23336596)

[3. Functionality 2](#_Toc23336597)

[3.1 Functional Requirements 2](#_Toc23336598)

[3.2 Non-Functional Requirements 2](#_Toc23336599)

[3.3 Assumptions 2](#_Toc23336600)

[4. System Architecture 2](#_Toc23336601)

[5. Use Cases 2](#_Toc23336602)

[5.1 Sample Use Case Name Here 2](#_Toc23336603)

[6. Graphical User Interfaces 2](#_Toc23336604)

[7. Database Design (*if required*) 2](#_Toc23336605)

[7.1 ER Diagram 2](#_Toc23336606)

[7.2 Data Dictionary 2](#_Toc23336607)

[8. Risk Analysis 2](#_Toc23336608)

[9. System Requirements 2](#_Toc23336609)

[9.1 Hardware Requirements 2](#_Toc23336610)

[9.2 Software Requirements 2](#_Toc23336611)

[10. References 2](#_Toc23336612)

[11. Appendix 2](#_Toc23336613)

[12. Formatting Guidelines (This chapter is not part of FS) 2](#_Toc23336614)

[12.1 Tables and Figure 2](#_Toc23336615)

[12.2 Equations 2](#_Toc23336616)

[12.3 Header/Footer 2](#_Toc23336617)

[12.4 References 2](#_Toc23336618)

List of Tables

[Table 2: This is Sample table caption 2](#_Toc19700198)

List of Figures

Figure 1: List of Styles 2

# Introduction

Internet of Things has become a giant field over the last decade and it has a wide range of applications like smart home system and smart security system. But due to this technology boom, IoT devices are becoming vulnerable to the attacks like Buffer Overflow, XSS, SQLI, etc. And with very basic malfunctioning of inputs, behavior of these devices can be altered or controlled.

To address this issue, a testing technique referred as “Fuzzing” will be used in this project. Fuzzingis a software technique that is used to uncover variety of issues such as coding errors and system crashes and so forth using malformed/semi-malformed data as program input in an automation fashioned way. It has two types: Mutational and Generational. In mutational fuzzing, input seeds are mutated with random characters, whereas generational fuzzing is an intelligent approach in which understanding in format and structure of input seeds is involved and generates input from scratch based on the specification or format.

Our Project focuses on monitoring and detecting security bugs and memory leaks in IoT applications and protocols by building a tool which applies fuzzing technique to crash system. This tool will apply either mutational or generational technique to alter the behavior of system. Also, it will provide an analysis report after fuzzing on a system. This tool will be an open-source tool and will be available for public use.

## Purpose of this Document

The purpose of this document is to provide an overview of IoT Fuzzer tool. It contains all the necessary information about the project, purpose, intended audience, domain overview, architectural design, and graphical user interface and system requirements. The user characteristics, functional and non-functional requirements and other necessary details are also written in this document.

## Intended Audience

The primary audience of this document is the project evaluation panel of our university. Other categories of audience may include IoT devlopers, software devleoper and security engineers. In general any researcher or developer who wants to explore vulnerabilities and flaws in IoT devices.

## Definitions, Acronyms, and Abbreviations

* IoT developers: An IoT developer is one popular role within the larger IoT field, focusing primarily on the creation of software that allows products to function and connect to other devices
* Software developer: A developer is the key individual behind all software applications who is responsible for creating software and applications. He or she writes, debugs and executes the source code of a software application.
* Security Engineer: Security engineers develop and supervise data and technology security systems to help prevent breaches, taps, and leaks associated with cybercrime.
* Evaluation Panel: An evaluation panel established by the concerned university department for the purpose of evaluating project deliverables. They are a group of people that have the necessary technical knowledge required to evaluate a certain project.

# General Description

## User Characteristics

**3.1.1 Security Engineer**

Security Engineers/Testers will make up most of the users of the application. IoTfuzzer will provide a complete “Fuzz testing or Fuzzing platform” for users, security engineers would be able to exploit different vulnerabilities and flaws in different IoT.

**3.1.1 Developers**

Developers would be able to use IoT fuzzer, for further extensions, addition of new protocols, addition of new devices and to spot bugs in their software.

## Domain Overview

Our system will provide an efficient monitoring tool which detects system crashes and bugs by applying the fuzzing technique. First user will have to connect to an app which can be done by providing IP and Port number of that app then user have to make a payload which can be custom made or user can use one of the built-in payloads and then set some fuzzing configurations such as setting depth etc. System can also provide an analysis report to user if user wants, otherwise, it will just show some metrics on runtime such as total crashes and number of cycles etc.

# Functionality

## Functional Requirements

* The system shall allow the user to use different Network Protocols including TCP, UDP, HTTP, WiFi, CoAP, and MQTT etc. to communicate with IoT devices.
* The system must generate log files for a fuzzing session.
* The system shall allow the user to perform mutation-based fuzzing on the IoT system.
* The system shall allow the user to upload seed from file
* The system shall allow the user to connect to IoT devices. For connection a user will provide target device’s IP and Port
* The application must be able to select optimal test cases from seed files, to select maximum code coverage in IoT device.
* The system shall allow the user to do generation-based fuzzing dependent on input structure of the IoT system.
* The system shall allow the user to save crash reports like buffer overflow, Dos and memory corruption (if any occurs during the fuzzing session).
* The system shall allow the user to send payload to target IoT device connected to the host system.
* The system shall allow the user to hook payload to specific protocol in the target IoT device.
* The system must be able to display current map coverage, mutations, and stage and cycle progress.
* The system shall allow the user to test connection with the target IoT device, whether device is receiving messages from the host systems or not.
* The system must be able to receive feedback from the target system, the response generated for the given input, any crashes or bugs spotted in the target system.
* The system must be able to save test cases that caused crash with unique identifiers.
* The system must show overall statistics for the test session.
* The system must display history of fuzzing, previous test done etc..
* The system must be able to run child processes on multiple cores, depending on fuzzing.

## Non-Functional Requirements

### Usability:

The interface shall be user friendly and easy to use. Simple English shall be used. There will be clear error message when the system does not respond or crashes, error message will provide directions what to do. Also, a walkthrough video shall be available for user to get guidance of how to use the system if they stuck at something.

### Performance:

The system shall make good use of processor cores and CPU cycles to optimize the fuzzing and no of tests per min. Also, system shall update all the plots after every second.

### Extensibility

The system shall use modular approach so that adding new module of a new feature will not cause any major problem with previously added modules

## Assumptions



# System Architecture

Our system will use client-server model. The client-server model describes how a server

provides resources and services to one or more clients. The reason to use client-server model

is because our system is not a shared resource, moreover , the system has to be platform

independent i.e it would work on both windows and Linux.

There would be three layers for the client-server architecture application layer, presentation

layer and server with a Fuzzing Engine.

**7.1 Presentation layer**

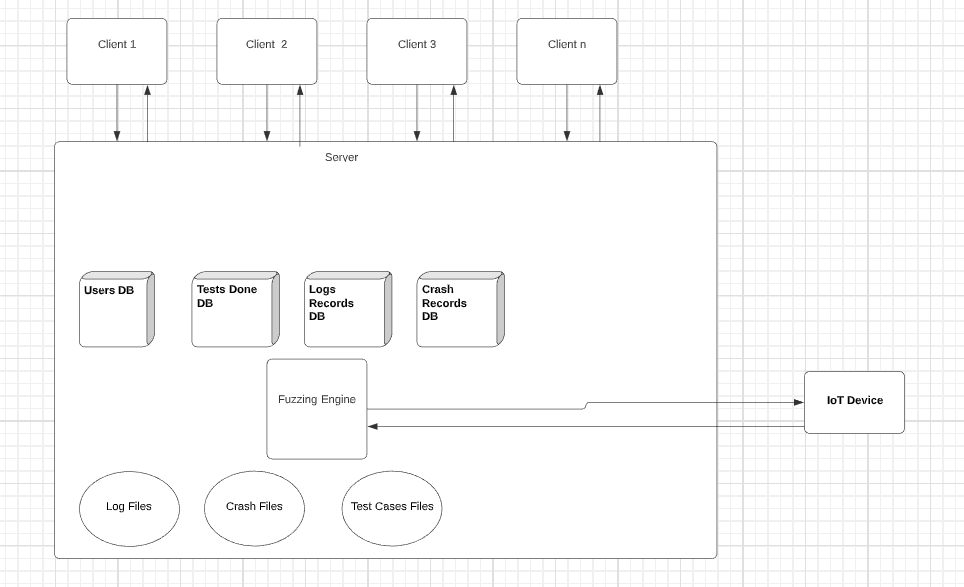
Presentation layer is the interface provided for user interaction.

**7.2 Application layer**

Application layer deals with the functionalities of the system.

**7.3 Database layer**

Database layer deals with all the databases of the system.



# 

# Use Cases

## Connect to an IoT Device

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Connect to an IoT Device | | |
| Actors | | User | | |
| Summary | | The user shall provide the target IoT device’s IP address and Port number and system will send packets to that IP and Port that | | |
| Pre-Conditions | | The user must be in the database records either added by any of the authorized users or added manually by a developer.  The user must not already be logged in. | | |
| Post-Conditions | | The user’s session is successfully established and shall be redirected to the home page. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the login page. | | 2 | The login page is displayed asking for email and password. |
| 3 | The user enters valid email and password. | | 4 | The system verifies the email and password, establishes a session for the user and redirects the user to the home page. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system responds with an error message: *Incorrect email or password entered.* |

## Make Custom Payload (Mutation)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Make Custom Payload | | |
| Actors | | User | | |
| Summary | | The user shall enter a string and other metrics such as ratio, length etc. and create a custom payload | | |
| Pre-Conditions | |  | | |
| Post-Conditions | | Custom payload pop-up screen will be closed. A custom payload is created and will appear on the fuzzing configuration screen. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user clicks on custom payload button. | | 2 | Custom payload pop-up screen opens |
| 3 | User enters the payload string | | 4 | Payload string is displayed on the text area of custom payload |
| 5 | User sets the parameters such as ratio, range etc. | | 6 | Parameters are set. |
| 7 | User clicks on create button | | 7 | Custom pop-up screen will be closed. And the custom payload is displayed on configuration screen |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system responds with an error message: *Incorrect email or password entered.* |

## Display Current System Status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Display Current System Status | | |
| Actors | | User | | |
| Summary | | The user after establishing connection and setting the configuration, when starts the fuzzing, then system will provide current statistics and process information. | | |
| Pre-Conditions | | Fuzzing is initiated | | |
| Post-Conditions | | System status will be displayed. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User requests for current system status. | | 2 | System displays currently running processes, fuzzing information and number of processes. |
| **Alternative Flow** | | | | |
| 1 | User requests for current system status. | | 2-A | The system responds with an error message: “Target device not found” |
| **Alternative Flow** | | | | |
| 1 | User requests for current system status. | | 2-B | The system responds with an error message: “Error getting system information” |

## Generate Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Generate Test Cases | | |
| Actors | | User | | |
| Summary | | The user shall provide a seed and system will create several test cases intelligently | | |
| Pre-Conditions | | Protocol is selected | | |
| Post-Conditions | | N-number of Test cases will be generated. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User provide an input seed | | 2 | System accepts seed. |
| 3 | User provide the number of test cases | | 4 | System displays the number of test cases on screen |
| 5 | User selects to generate the test cases. | | 6 | System generates the test cases with maximum code coverage. |
| **Alternative Flow** | | | | |
| 5 | User selects to generate the test cases. | | 6-A | The system responds with an error message: “Empty seed is provide” |

## Select a Built-in Input Seed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Select a Built-in Input Seed | | |
| Actors | | User | | |
| Summary | | The user shall select one of many built-in available input seed | | |
| Pre-Conditions | | Fuzzing category is selected | | |
| Post-Conditions | | Built-in Input Seed will be selected and displayed on the configuration screen. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user open configuration page | | 2 | Configuration page is opened |
| 3 | The user clicks on select a built-in input seed. | | 4 | A list of built-in input seeds will be displayed |
| 5 | User selects an input seed | | 6 | Input seed is selected and displayed on configuration screen |

## Pause Fuzzing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Pause Fuzzing | | |
| Actors | | User | | |
| Summary | | The fuzzing shall be paused after user commands it. | | |
| Pre-Conditions | | Fuzzing is initiated. | | |
| Post-Conditions | | Fuzzing shall be paused. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User initiates Fuzzing | | 2 | Fuzzing processes Start |
| 3 | User clicks pause button | | 4 | System shall pause the fuzzing |
|  |  | | 5 | System displays resume option and stop option. |

## Stop Fuzzing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Stop Fuzzing | | |
| Actors | | User | | |
| Summary | | The fuzzing shall be stopped after user commands it. | | |
| Pre-Conditions | | Fuzzing is initiated. | | |
| Post-Conditions | | Fuzzing shall be stopped and user shall be presented with save crash file, log file and analysis report options. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User initiates Fuzzing | | 2 | Fuzzing processes Start |
| 3 | User clicks stops button | | 4 | System shall stop fuzzing |
|  |  | | 5 | System displays options to save analysis report, crash file and log file. |

## View Fuzzing History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | View Fuzzing History | | |
| Actors | | User | | |
| Summary | | Displays previous fuzzing records | | |
| Pre-Conditions | | User is registered | | |
| Post-Conditions | | The user will be presented with history. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User Presses View history | | 2 | System get logs. |
|  |  | | 3 | System Displays History |

## Save Analysis Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Save Analysis Report | | |
| Actors | | User | | |
| Summary | | After performing fuzzing, system saves analysis report of fuzzing. | | |
| Pre-Conditions | | Fuzzing is finished | | |
| Post-Conditions | | An analysis report is downloaded on the user’s system | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks on save analysis report. | | 2 | A pop-up window appears in system. |
| 3 | User gives the path of download file. | | 4 | System Generates Crash Reports |
|  |  | | 5 | System Saves Crash Reports with unique id for every crash, crash details and test input. |
| **Alternative Flow** | | | | |
| 3 | User initiates Fuzzing | | 4-A | Target System is not connected/offline, System Displays “N/A for Target System”. |

## Search a Database Record

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Search a Database Record | | |
| Actors | | User | | |
| Summary | | User shall search a report or file from its database record. After querying through database system shall provide user with the result file or record that user has asked for | | |
| Pre-Conditions | | User is registered and user’s record is present in database | | |
| Post-Conditions | | Relevant database records are displayed to the system | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks on search a database record | | 2 | User is redirected to the search screen |
| 3 | User enters a valid search string | | 4 | System make a query of that search string. |
|  |  | | 5 | System will query that into database. |
|  |  | | 6 | System will display the results to the search screen |
| **Alternative Flow** | | | | |
| 3 | User does not enter a valid string | | 4-A | System responds with the message: *No records are to be found with this search string.* |

## Create Custom Payload (Generation Based)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Create Custom Payload (Generation based) | | |
| Actors | | User | | |
| Summary | | The user will select a protocol, enters a string and parameters such as range, ratio to make a custom payload that fits it in that data model. | | |
| Pre-Conditions | | Generation based fuzzing category is selected. | | |
| Post-Conditions | | A custom payload will be generated and displayed on configuration screen. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks on custom payload | | 2 | A pop-up window is opened. |
| 3 | User clicks on select protocol button | | 4 | A protocol list will be displayed |
| 5 | User selects protocol | | 5 | Protocol is selected |
| 6 | User enters a string in text area | | 7 | A string is displayed on text area |
| 8 | User sets the other custom parameters such as range, ratio etc. | | 9 | Parameters are set |
| 10 | User clicks on create | | 11 | Pop-up window gets closed. A custom payload is created and displayed on screen. |
| **Alternative Flow** | | | | |
| 10 | User clicks on create | | 11 | System generates an error displaying that payload cannot be fit in the format. |

## Fork Child Fuzzer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Fork Child Fuzzer | | |
| Actors | | User, Host System(system) | | |
| Summary | | Fork Multiple child Processes (Fuzzers) on different CPU cores. | | |
| Pre-Conditions | | Start Fuzzing | | |
| Post-Conditions | | None | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User initiates Fuzzing | | 2 | Fuzzing processes Start |
| 2 | User Specifies Number of Processes/CPU’s for usage | | 3 | System initiates child process accordingly. |
|  |  | | 4 | Child process fuzz target system same as master process. |
| **Alternative Flow** | | | | |
| 3A | User initiates Fuzzing | | 4-A | Target System is not connected/offline, System Displays “N/A for Target System”. |

## Start Fuzzing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Start Fuzzing | | |
| Actors | | User | | |
| Summary | | Start Fuzzing For Specific IoT device using given protocol | | |
| Pre-Conditions | | Fuzzing configuration are set | | |
| Post-Conditions | | Fuzzing is started and system statistics will be displayed on screen. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User press start fuzzing button | | 2 | System starts fuzzing |
| **Alternative Flow** | | | | |
| 1 | User press start fuzzing button | | 2-A | System generates an error saying that configuration settings are not valid. |

## Mutate Input

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Mutate Input | | |
| Actors | | Host System | | |
| Summary | | Mutate Input continuously during Fuzzing | | |
| Pre-Conditions | | Fuzzing is initiated | | |
| Post-Conditions | | None | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User initiates Fuzzing | | 2 | Fuzzing processes Start |
|  |  | | 3 | System Mutates Fuzzing Input By either Bit Flips, Genetically or other ways |

## Upload Seed File

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Upload Seed File | | |
| Actors | | User | | |
| Summary | | User shall upload a payload from a file. | | |
| Pre-Conditions | | Fuzzing Category is selected. | | |
| Post-Conditions | | A seed is displayed on the configuration screen | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks on upload seed from file button | | 2 | A pop-up window appears |
| 3 | User selects a file from the from window | | 4 | A file is selected. |
| 5 | User clicks save button | | 6 | A seed is displayed on the configuration screen. |
| **Alternative Flow** | | | | |
| 7 | User uses drag and drop method to upload a file. | | 8 | A file is selected |
| 9 | User clicks save button | | 10 | A seed is displayed on the configuration screen. |
| **Alternative Flow** | | | | |
| 5 | User clicks save button | | 6-A | System Generates an error displaying that the wrong file format is selected |
| **Alternative Flow** | | | | |
| 5 | User clicks save button | | 6-B | System Generates an error displaying that the file cannot be opened because it is password protected |
| **Alternative Flow** | | | | |
| 5 | User clicks save button | | 6-C | System Generates an error displaying that the file is corrupted. |

## Save Log File

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Save Log File | | |
| Actors | | User | | |
| Summary | | Saves Log Files of Tests run. | | |
| Pre-Conditions | | Fuzzing is initiated. | | |
| Post-Conditions | | A log file is saved in user’s machine. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks on save log file button. | | 2 | A pop-up window. |
| 3 | User selects the saving path. | | 4 | A saving path is selected. |
| 5 | User press save. | | 6 | File starts downloading. |
|  |  | | 7 | File gets saved on the specified path. |

## Test Connection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Test Connection | | |
| Actors | | User | | |
| Summary | | This shall Check Whether a Connection is established with IoT device on specified Protocol and address | | |
| Pre-Conditions | | IP address and Port number of target device should be given in configuration section. | | |
| Post-Conditions | | A response message from target device will be received. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User Presses Test Connection | | 2 | System sends packet for request |
| 3 |  | | 3 | Target Systems replies with response message. |
| **Alternative Flow** | | | | |
| 1 | User Presses Test Connection | | 3-A | System generates an error displaying that IP address or Port number is invalid. |
| **Alternative Flow** | | | | |
| 1 | User Presses Test Connection | | 3-B | System generates an error displaying that packet is not sent due to internet traffic. |

## Sign-up

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Sign-up | | |
| Actors | | User | | |
| Summary | | The user shall provide their credentials on the sign form and after successful validation and email confirmation, redirect the user to the home page. | | |
| Pre-Conditions | | The user email must be unique. User must not be in the database records. | | |
| Post-Conditions | | A user account is created and saved in database. The user’s session is successfully established and shall be redirected to the home page. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the sign-up page. | | 2 | The sign-up page is displayed asking for user’s credentials. |
| 3 | The user enters valid credentials | | 4 | The system validates the credentials |
|  |  | | 5 | The system sends a confirmation email to user’s email address. |
| 6 | The user confirms the confirmation email. | | 7 | System redirects the user to the homepage. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid credentials. | | 4-A | The system responds with an error message: *Invalid credentials are entered.* |
| **Alternative Flow** | | | | |
| 6 | The user don’t confirms the email | | 7-A | The system responds by displaying a message on sign-up page: *Kindly confirm email.* |

## Logout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Logout | | |
| Actors | | User | | |
| Summary | | The user session will be finished after user commands the system to logout. | | |
| Pre-Conditions | | The user must be logged in. | | |
| Post-Conditions | | User will be redirected to login screen. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user clicks on logout button. | | 2 | The system will finish user’s session |
|  |  | | 3 | The system will redirect the user to login page |

## Display Mutators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Display Mutators | | |
| Actors | | User | | |
| Summary | | The system should display the mutators used in the current fuzzing session. | | |
| Pre-Conditions | | The user must have done Fuzzing at least once | | |
| Post-Conditions | | The user is displayed mutators used in that session. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User Presses display mutators option | | 2 | System displays list of mutators used during session, along with count, iteration and faults |

## Display Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Display Elements | | |
| Actors | | User | | |
| Summary | | The system should display the Elements that caused crashes. | | |
| Pre-Conditions | | The user must have done Fuzzing at least once | | |
| Post-Conditions | | The user is displayed mutators used in that session. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User Presses display Elements option | | 2 | System displays list of Elements response, parameters, iteration and faults. |

## Login

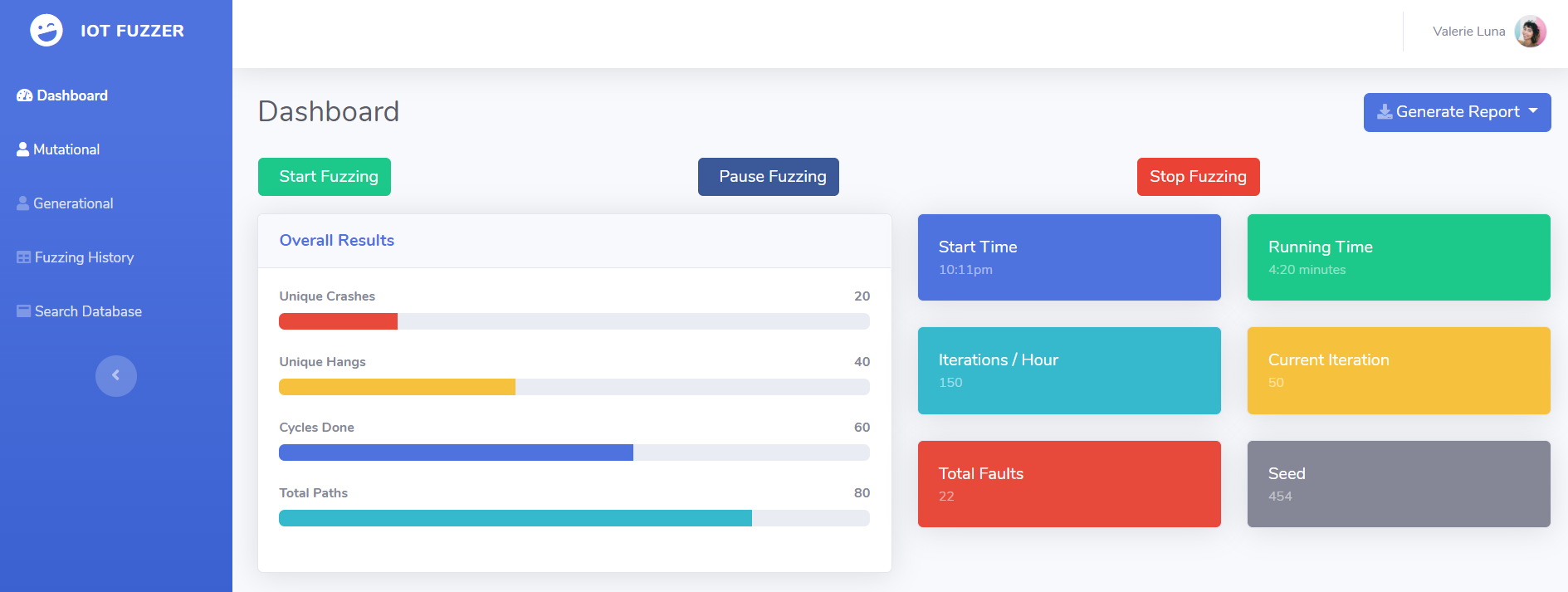
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Login | | |
| Actors | | User | | |
| Summary | | The user shall provide their email and password on the login form and after successful verification, redirect the user to the home page. | | |
| Pre-Conditions | | The user must be in the database records either added by any of the authorized users or added manually by a developer.  The user must not already be logged in. | | |
| Post-Conditions | | The user’s session is successfully established and shall be redirected to the home page. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the login page. | | 2 | The login page is displayed asking for email and password. |
| 3 | The user enters valid email and password. | | 4 | The system verifies the email and password, establishes a session for the user and redirects the user to the home page. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system responds with an error message: *Incorrect email or password entered.* |

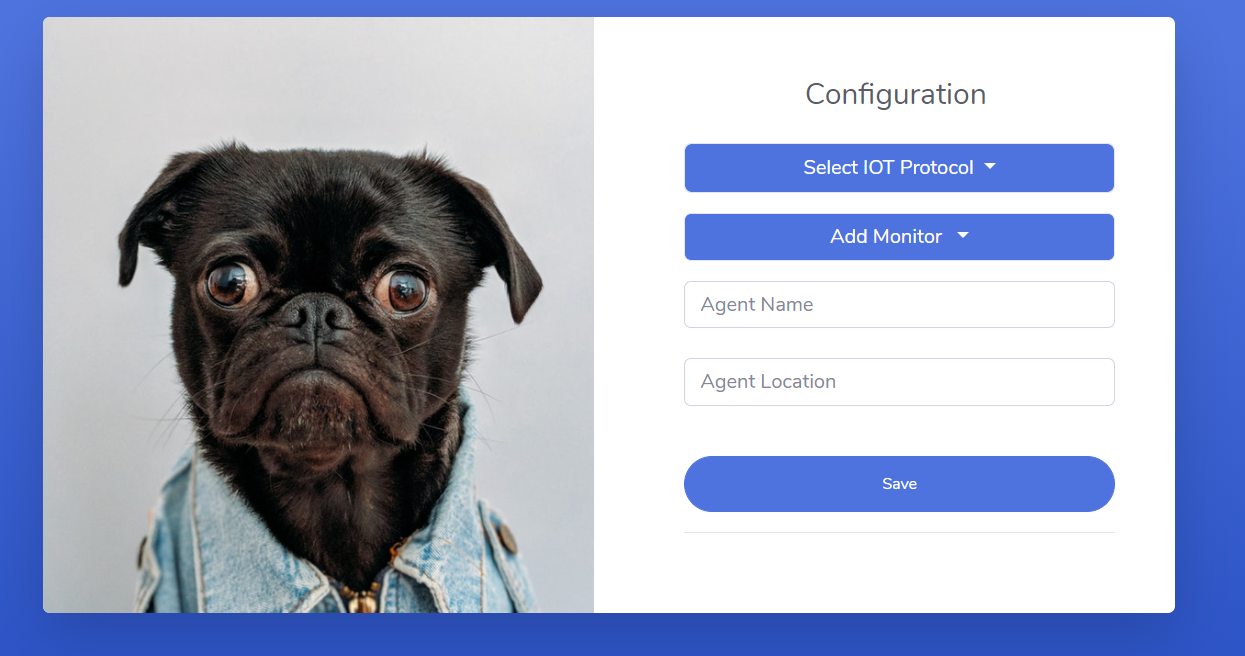
# Graphical User Interfaces

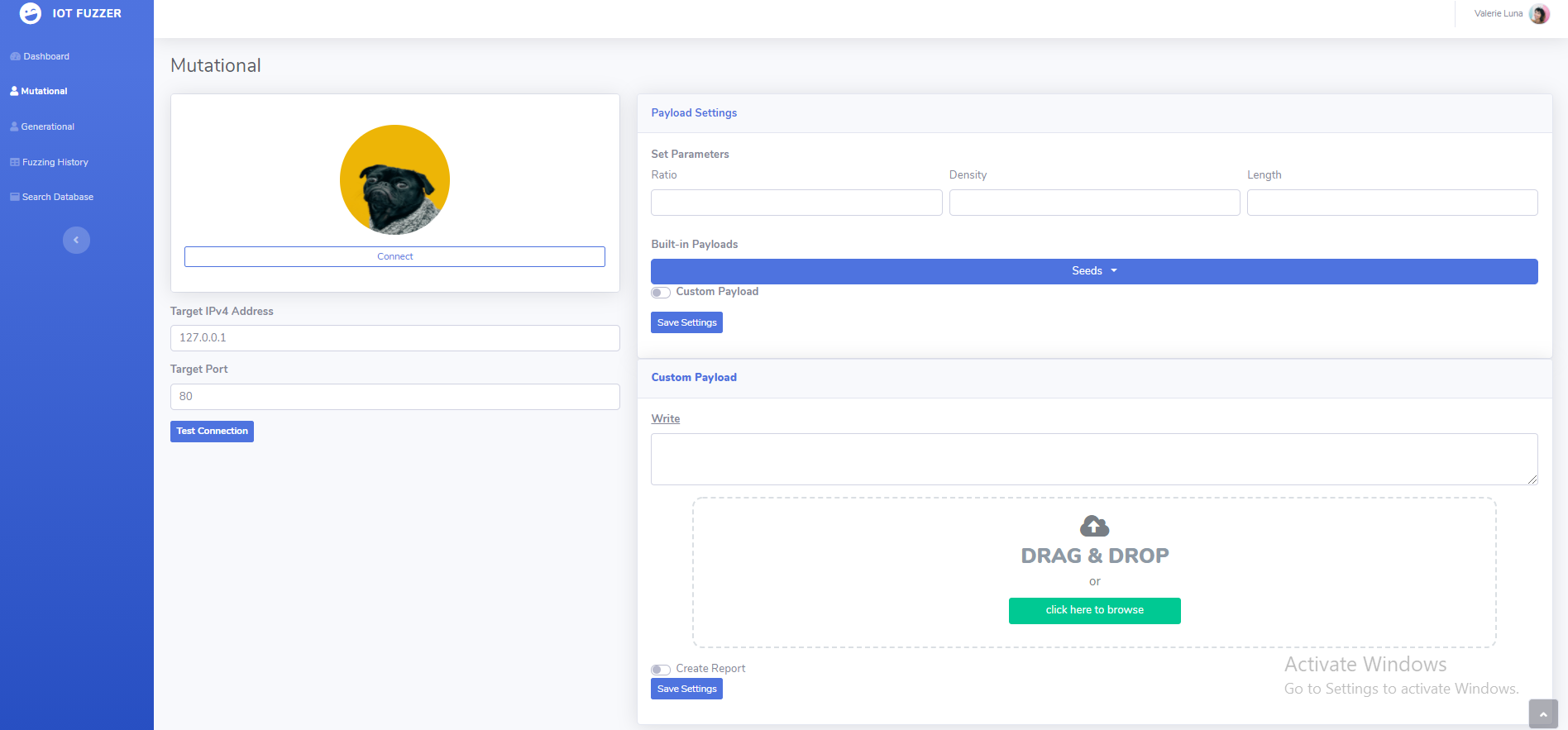
**Figure 1 Fuzzing Stats**

# 

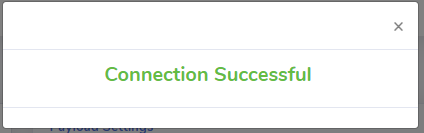
# Figure 2 Main Page



**Figure 3 Configure connection**

**Figure 4 Mutational Fuzzing** 

**Figure 5 Connection Sucess**



# Database Design (*if required*)

## ER Diagram

## Data Dictionary

Table 1: The table for User

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Email | Nvarchar(50) | Stores email of user, primary key |
| First Name | Nvharchar(50) | Stores first name of user |
| Last Name | Nvarchar(50) | Stores last name of user |
| Password | Nvarchar(50) | Stores password of user |

Table 2: The table for Test Cases

The table shows the attributes and their data types required for keeping record in the database of the test cases

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Test\_id | Int | Primary key for identifying test |
| Email | Nvharchar(50) | Email of user who has done test , as For |
| Process | Int | No of processes used for fuzzing |
| deviceId | Nvarchar(50) | Device on which fuzzing/testing is done |

Table 3: The table for Crash Report

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Crash\_id | int | Stores id for crash file , primary key |
| fileName | Nvharchar(50) | Stores crash file name |
| Test id | int | Foreign key, identifies crash file to test number |

Table 4: The table for Crash Report

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| log\_id | Int | Stores id for log file , primary key |
| Log\_fileName | Nvharchar(50) | Stores Log File name |
| Test id | Int | Foreign key, identifies log file to test number |

Table 5 The table for Test Cases report

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Crash\_id | Int | Stores id for crash file , primary key |
| fileName | Nvharchar(50) | Stores test cases file name |
| Test id | Int | Foreign key, identifies crash file to test number |

Table 6: The table for Test Results

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| No of Hangs | Int | Stores number of hangs occurred during fuzzing |
| No of Crashes | Int | Stores number of crashes occurred during fuzzing |
| No of Cycle | Int | Stores number of cycles occurred during fuzzing |
| Test\_id | Int | Foreign key, identifies item for specific test case |

Table 7: The table for Findings

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Cycles executed | Int | Stores number of cycles executed. |
| Speed/cycle | Int | Stores execution speed |
| Mutations done | Int | Stores number of mutation |
| Test\_id | Int | Foreign key, identifies item for specific test case |

Table 8: The table for Mutations

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Value | Nvarchar(50) | Value of payload |
| Type | Nvarchar(50) | Type of payload value |
| Fault | Int | Number of faults caused |
| Test\_id | Int | Foreign key, identifies item for specific test case |

Table 9: The table for Elements

The table shows the attributes and their data types required for keeping record in the database of the User

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| Iteration | Int | Number of iterations |
| Element | Nvarchar(50) | Element that caused fault. |
| Fault | Int | Number of faults caused |
| Test\_id | Int | Foreign key, identifies item for specific test case |

# Risk Analysis

List and explain the risks that maybe encountered during the project. For e.g.: technical risks, business risks etc.

# 

# System Requirements

List the hardware and software requirements that will be required to develop and deploy the project.

## Hardware Requirements

## Software Requirements

# References

List all important sources of information which have been consulted for this project

# Appendix

This section should include all supporting information from the project that was not included in the body of the report.  You should include surveys, complex statistical calculations, certain detailed tables and other such information in an appendix.  The information presented in this section is important to support the work presented in the body of the report but would make it more difficult to read and understand if presented within the body of the report.

Cite the appendix items in the report narrative (write "see Appendix A") and organize appendices (e.g., Appendix A, Appendix B,

Any tables, figures, forms, or other materials that are not totally central to the analysis but that need to be included are placed in the Appendix.

# Formatting Guidelines (This chapter is not part of FS)

This document also serves as style guide for final year project reports. In order to give a similar high quality appearance to all final year software project reports this template uses a collection of predefined Microsoft Word formatting styles. **These styles should be used without modification or replacement.** Following styles have been used

* **Title** – the main title style
* **Title2** – the subtitle style
* **Heading 1, Heading 2, Heading 3** – styles for different levels of section headings
* **Body Text** – style for paragraphs
* **Caption** – the style for a figure or table caption
* **Table Description** – the style for description of table
* **Figure Description** - the style for description of figure
* **Code** – the style for program source code
* **Table Header Row** – Style for the header row of table
* **Table Grid** – the style for the data rows in the tables
* **Reference** – The style for reference
* **Bullets** – The style for this list
* **Numbered** – Style for numbered list

Press Ctrl+Shift+S to see list of styles mentioned above. Figure 1 shows the Apply Style window displaying the list of styles. Select any text then press Ctrl+Shift+S, the Apply Style window will show you the current style applied on that text and if required, you can change the style by selecting any other style from the “Style Name” dropdown.

This is brief description of above figure.

Figure 1: List of Styles

## Tables and Figure

Tables and figures should be centered horizontally. The caption button should be used to insert caption for both the figures and tables. All figures and tables must be numbered properly. Always refer to tables and figures according to their numbers. A table or figure can be cited as follows: ‘see Table1’ or ‘as shown in Table1’. The caption of table should be centered above the table and figure caption should be centered below the figure. Place the tables/figures close to their reference. Use “Table Header Row” and ‘Table Grid’ style for table’s header and data rows respectively. It is compulsory to provide brief description of table/figure after its caption. Styles for table and figure descriptions are “Table Description” and “Figure Description” respectively.

Table 1: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

## Equations

Use equation editor to write equations in this report. Use last button of the custom tool bar to invoke equation editor. Similar to tables and figures, equations should also be aligned centered horizontally. Number all important equations and insert them in parenthesis. Below is a sample equation and its reference number. An equation can be referenced like this: ‘it is clear from (1)’.

 (1)

## Header/Footer

Notice the headers in this document, before Introduction (i.e. the main content of this document) page numbers are in roman numerals. The page numbers of the actual content start with Arabic numerals i.e. 1, 2, 3 and so on. All the odd numbered pages contain title of your project while the even numbered pages contain the section heading (i.e. chapter’s name) in the headers.

## References

Always refer to the source of information by inserting the reference number in square brackets like this [5]. The reference numbers can either be added at the end of the sentence or within the sentence without changing the punctuation of sentence. A reference can also be cited as follows: ‘as Ruskey [2] mentioned’. List each source only once on your reference page.

**Following is a list of sample reference for various typed of sources in IEEE format.**

**Book**

1. P.M. Morse and H. Feshback, *Methods* of *Theoretical Physics*. New York: McGraw Hill, 1953.

**Journal Article**

1. S.K. Kenue and J.F. Greenleaf, “Limited angle multifrequency diffiaction tomography,” *IEEE Trans. Sonics Ultrason*., vol. SU-29, no. 6, pp. 213-2 17, July 1982.

**Dissertation or thesis**

1. B. Tsikos, “Segmentation of 3-D scenes using multi-modal interaction between machine vision and programmable mechanical scene manipulation,” Ph.D. dissertation, Univ. of Pennsylvania, BCE Dept., Philadelphia, 1987. [Add if applicable: University Microfilms, Inc., University of Michigan, Ann Arbor, Michigan.]

**Proceedings paper**

1. R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman, “An overview of AL, programming system for automation,” in *Proc. Fourth Int. Joint Conf Artif. Intell*., pp. 758-765, Sept. 3-7, 1975.

**Newspaper article**

1. “Technology threatens to shatter the world of college textbooks, *The Wall Street Journal*, vol 91, pp. Al, A8, June 1, 1993.

**Technical Report**

1. R. Cox and J. S. Turner, “Project Zeus: design of a broadband network and its application on a university campus,” Washington Univ., Dept. of Comp. Sci., Technical Report WUCS-91-45, July 30, 1991.

**Software**

1. M. Janzen, *Instant Access Accounting*. Computer software. Nexus Software, Inc IBM-PC, 1993.

**World Wide Web** (give author and title if named)

1. Fuminao Okumura and Hajime Takagi, “Maglev Guideway On the Yamanashi Test Line,” *http://www.rtri.or.jp/rd/maglev2/okumura.html*, October 24, 1998.
2. “AT&T Supplies First CDMA Cellular System in Indonesia,” *http://www.att.com/press/1095/951011.nsa.html*, Feb 5, 1996.