# Acknowledgement

I would like to take this opportunity to express my gratitude to everyone who supported me throughout this project. It is a privilege to thank project supervisor, Dr. Basanta Joshi for his invaluable yet constructive criticisms, ambitious guidance and countless advices on my project work. I would also like to thank our IT faculty leader (Vice Dean) Mr. Arun Lal Joshi for his support during this project work. Finally, I would grab this opportunity to thank my family, colleagues and teachers for their direct or indirect suggestions and support during research and development for project.

# Abstract

Development of an CV (resume) portal requires selection of right database technology. This project proposes to use NoSQL while keeping SQL as alternative solution. It is essential to perform both practical as well as theoretical research on NoSQL technology to understand suitability and feasibility for resume portal project. A thorough review of traditional relational database is done to learn their limitations and strengths. These findings are used for critically comparing with strengths and limitations of NoSQL technology. Analysis, planning, design, development and testing portal is carried out within this report while also evaluating developed portal system and overall project.

# Table of Contents

[i. Acknowledgement 1](#_Toc477107811)

[ii. Abstract 2](#_Toc477107812)

[iii. Table of Contents 3](#_Toc477107813)

[1. Introduction 8](#_Toc477107814)

[1.1. Background/Problem Statement 8](#_Toc477107815)

[1.2. Aims and Objectives 8](#_Toc477107816)

[1.3. Project Scope 9](#_Toc477107817)

[1.3.1. High Level Project Specification 9](#_Toc477107818)

[1.3.2. Boundaries 10](#_Toc477107819)

[1.4. Strategy 10](#_Toc477107820)

[1.5. Deliverables 10](#_Toc477107821)

[1.6. Acceptance Criteria 11](#_Toc477107822)

[1.7. Constraints 11](#_Toc477107823)

[2. Review of Literature 12](#_Toc477107824)

[2.1. Relational Database 12](#_Toc477107825)

[2.2. Limitations with Relation Database 13](#_Toc477107826)

[2.2.1. Schema Change 13](#_Toc477107827)

[2.2.2. Complex Join 13](#_Toc477107828)

[2.2.3. Slow Query Performance 13](#_Toc477107829)

[2.3. NoSQL misconception 13](#_Toc477107830)

[2.4. Features 14](#_Toc477107831)

[2.4.1. Schema agnostic 14](#_Toc477107832)

[2.4.2. Nonrelational 14](#_Toc477107833)

[2.4.3. Highly distributable 14](#_Toc477107834)

[2.5. Types/NoSQL Classification 15](#_Toc477107835)

[2.5.1. Key-Value Store 15](#_Toc477107836)

[2.5.2. Column family stores 15](#_Toc477107837)

[2.5.3. Graph Store 16](#_Toc477107838)

[2.5.4. Document Database 16](#_Toc477107839)

[2.6. Why Document Type 17](#_Toc477107840)

[2.7. MongoDB 17](#_Toc477107841)

[2.7.1. Why MongoDB 18](#_Toc477107842)

[2.8. MongoDB Terminology and Concepts 19](#_Toc477107843)

[2.9. Comparison between Relational and NoSQL 21](#_Toc477107844)

[2.9.1. Application Development 21](#_Toc477107845)

[2.9.2. Data Storage Model 22](#_Toc477107846)

[2.9.3. Schema 23](#_Toc477107847)

[2.9.4. Vendors 23](#_Toc477107848)

[2.9.5. Eco System 24](#_Toc477107849)

[2.9.6. Normalization in SQL and De-normalization in NoSQL 24](#_Toc477107850)

[2.9.7. Data Integrity 24](#_Toc477107851)

[2.9.8. Transaction Model 24](#_Toc477107852)

[3. Review of Technologies 26](#_Toc477107853)

[3.1. Data Access [CRUD] 26](#_Toc477107854)

[3.2. MongoDB in web application development 27](#_Toc477107855)

[3.2.1. MongoDB C# driver 27](#_Toc477107856)

[3.3. Practical Comparison 27](#_Toc477107857)

[4. Proposed Methodologies 28](#_Toc477107858)

[4.1. Project Life Cycle Model 28](#_Toc477107859)

[4.2. Model View Controller (MVC) Pattern 28](#_Toc477107860)

[5. Product Design 30](#_Toc477107861)

[5.1. Use Case Diagram 30](#_Toc477107862)

[5.2. Flow Chart 30](#_Toc477107863)

[5.3. Data Flow Diagram 35](#_Toc477107864)

[5.4. Sequence Diagram 35](#_Toc477107865)

[5.5. Context Diagram 37](#_Toc477107866)

[5.6. Data Dictionary 37](#_Toc477107867)

[5.7. Functional Requirements 38](#_Toc477107868)

[6. Product Implementation and Testing 40](#_Toc477107869)

[6.1. Implementation 40](#_Toc477107870)

[6.1.1. MongoDB Implementation 40](#_Toc477107871)

[6.1.2. MVC Pattern 40](#_Toc477107872)

[6.1.3. Service Layer 41](#_Toc477107873)

[6.1.4. Sending Email 42](#_Toc477107874)

[6.1.5. Publishing 42](#_Toc477107875)

[6.2. Testing 42](#_Toc477107876)

[6.2.1. Unit Testing 43](#_Toc477107877)

[6.2.2. Compatibility Testing 44](#_Toc477107878)

[6.2.3. Validation Testing 45](#_Toc477107879)

[7. Product and Project Evaluation 46](#_Toc477107880)

[7.1. Product Evaluation 46](#_Toc477107881)

[7.1.1. Strengths 46](#_Toc477107882)

[7.1.2. Limitations 46](#_Toc477107883)

[7.1.3. Recommendations 47](#_Toc477107884)

[7.2. Project Evaluation 48](#_Toc477107885)

[7.2.1. Limitations 49](#_Toc477107886)

[8. Summary 50](#_Toc477107887)

[9. Bibliography 51](#_Toc477107888)

[10. Appendices 53](#_Toc477107889)

[Appendix A 53](#_Toc477107890)

[Literature Review Images 53](#_Toc477107891)

[Technical Review Images 55](#_Toc477107892)

[Compatibility Test 56](#_Toc477107893)

[Validation Test Images 59](#_Toc477107894)

[Appendix B 61](#_Toc477107895)

[Feasibility Study 61](#_Toc477107896)

[Appendix C 64](#_Toc477107897)

[Implementations Images 64](#_Toc477107898)

[Appendix D 66](#_Toc477107899)

[Meeting Record Sheets 66](#_Toc477107900)

[Appendix D 73](#_Toc477107901)

[Git Repository Evidence 73](#_Toc477107902)

[Appendix E 74](#_Toc477107903)

[Appendix F 77](#_Toc477107904)

[Figure 1Use case diagram for resume portal 30](#_Toc477107570)

[Figure 2 Flowchart for resume portal (User registration) 31](#_Toc477107571)

[Figure 3 Flow chart for resume portal (User Login) 32](#_Toc477107572)

[Figure 4 Flowchart for resume portal (resume creation) 33](#_Toc477107573)

[Figure 5 Flowchart for resume portal (find resume) 34](#_Toc477107574)

[Figure 6 Data Flow Diagram for resume portal 35](#_Toc477107575)

[Figure 7Sequence Diagram for resume portal (registration) 35](#_Toc477107576)

[Figure 8Sequece Diagram for resume portal (login) 36](#_Toc477107577)

[Figure 9 Sequence Diagram for resume portal (resume form submit) 36](#_Toc477107578)

[Figure 10 Sequence Diagram for Resume portal (resume search) 36](#_Toc477107579)

[Figure 11 Sequence Diagram for resume portal (change password) 37](#_Toc477107580)

[Figure 12 Context Diagram for resume portal 37](#_Toc477107581)

[Figure 13 Model in MVC 40](#_Toc477107582)

[Figure 14 Controller in MVC 40](#_Toc477107583)

[Figure 15 View in MVC 41](#_Toc477107584)

[Figure 16 business logic for insert/update resume 41](#_Toc477107585)

[Figure 17Relational Database Structure 53](#_Toc477107586)

[Figure 18 Basic Structure of Modified Waterfall model 53](#_Toc477107587)

[Figure 19 Complex SQL database structure 54](#_Toc477107588)

[Figure 20 Key Value Database structure 54](#_Toc477107589)

[Figure 21 Setting up new connection in robomongo 55](#_Toc477107590)

[Figure 22Downloading MongoDB in Nuget 55](#_Toc477107591)

[Figure 23using namespace to use MongoDB driver 55](#_Toc477107592)

[Figure 24 resume portal in chrome browser 56](#_Toc477107593)

[Figure 25 resume portal in Opera browser 56](#_Toc477107594)

[Figure 26 resume portal in Mozilla browser 57](#_Toc477107595)

[Figure 27 resume portal in mobile browser 58](#_Toc477107596)

[Figure 28 resume portal in Tab browser 59](#_Toc477107597)

[Figure 29 login form validation test 59](#_Toc477107598)

[Figure 30registration form validation test 60](#_Toc477107599)

[Figure 31registration form password validation test 60](#_Toc477107600)

[Figure 32Downloading MongoDB 64](#_Toc477107601)

[Figure 33 Creating MongoDB instance as windows Service 65](#_Toc477107602)

[Figure 34MongoDB instance as windows service 65](#_Toc477107603)

[Figure 35Meeting record sheet 1 66](file:///D:\The%20British%20College\Project%20Production\PP%20Assignment\Final%20Report.docx#_Toc477107604)

[Figure 36Meeting Record sheet 2 67](file:///D:\The%20British%20College\Project%20Production\PP%20Assignment\Final%20Report.docx#_Toc477107605)

[Figure 37Meeting Record sheet 3 68](file:///D:\The%20British%20College\Project%20Production\PP%20Assignment\Final%20Report.docx#_Toc477107606)

[Figure 38Meeting Record sheet 5 69](file:///D:\The%20British%20College\Project%20Production\PP%20Assignment\Final%20Report.docx#_Toc477107607)

[Figure 39Meeting Record sheet 5 70](file:///D:\The%20British%20College\Project%20Production\PP%20Assignment\Final%20Report.docx#_Toc477107608)

[Figure 40Meeting Record sheet 6 71](#_Toc477107609)

[Figure 41Meeting Record sheet 7 72](file:///D:\The%20British%20College\Project%20Production\PP%20Assignment\Final%20Report.docx#_Toc477107610)

[Figure 42 Version Control client (Source Tree) 73](#_Toc477107611)

[Figure 43 Version Control Server (Bit Bucket) 73](#_Toc477107612)

# Introduction

## Background/Problem Statement

Having a good CV (abbreviated from **Curriculum Vitae**) is one of the fundamental requirement for making good imprint on possible employers. CV basically is a document that summarizes person’s skillsets, professional and educational experiences. Good cv boosts candidate’s opportunity to get called on an interview. People looking for job should carry at least one good cv that can highlight their skills, education, portfolio, experiences etc. It is not necessary that everyone can write a good cv for themselves. Hence, they are often found looking for right templates to prepare CV. Mcgee (2003) highlight the importance of good cv by noting that not only job seeker requires a good cv but students, consultants, graduates or simply those who just wants to change their career also needs good cv.

Preparing a CV can be difficult process that can require a lengthy time. Finding right template, preparing cv, converting them as PDF file then sending them to potential employer as an email attachment is basic process of applying for a job position. This seems simple but not everyone can enjoy this lengthy process specially when they have to apply to multiple companies. Hence there is need of an automation system for managing individual’s CV (**resume** in some countries).

An online web application system that could build cv for user automatically based on provided information can make things a lot easier. Application should not only create cv but should also user to share and print them. Employer should able to use this application to find potential employees with skill set their organization requires. This project would carry out research and development of an online resume portal product.

## Aims and Objectives

The core aim of this project would be research and development of online resume portal within the given resources while also accessing the use of NoSQL database technology as database solution for the product. This project would have following key aims and objective:

* Technical as well as literature based research on NoSQL technology for implementation in resume portal
* Define specification and analyze requirements for resume portal application
* Design product using UML diagrams such as use case, context, sequence etc.
* Use C# MongoDB driver to design and develop MongoDB database solution for CV project.
* Plan and design User interfaces using client side programming such as HTML5, CSS3, bootstrap and JQUERY etc.
* Implement authentication module in cv project using password encryption
* Use asp.net MVC framework to develop resume profile module to develop and manage resumes
* Complete development of other functional requirements for the resume portal application
* Carry out various testing such as compatibility testing, unit testing, data load testing
* Evaluate the product, usefulness of research and project as a whole

## Project Scope

The scope of the resume portal project is to research, analyze, plan and develop an online resume portal system which can automate the resume creation based on the information provided by user. Product would involve various modules such as authentication and authorization, setting, resume management and email feature. Product will be a browser based web application. Product will have a website for allowing to register and login into the system while home page of website also provide summative information about users of product. Registered users would be able to log into the system or ask for password through email if they forget them. Login into the system with right credentials would land user on a dashboard. This dashboard would contain links to various modules of product such as profile management, privacy management, resume creation or search other resumes based on their name, jobs, educations of skill. The scope of this document is to gather information for requirement, plan, design and implement cv portal.

### High Level Project Specification

The resume portal project has been approved after the submission of initial project plan. Project has identified following high level specifications:

* Resume portal system would research and implement MongoDB as its NoSQL database solution
* Resume portal would have sign up and login system and each password must be saved in encrypted form
* User would have ability to reset password via email and would receive email notification after registration
* Project development would be carried out using ASP.Net MCV framework.
* Product would enable user to automate cv development by providing personal information and education, job history, skill, language details.
* User would be able to find resumes based on name, education skills or keywords
* Product would allow user to print resume
* User would have ability to manage privacy of cv. They should be able to manage what information to show to public and what not to show.
* System should allow to save multiple resumes.

### Boundaries

This project includes requirement gathering, planning, designing and development (actual coding) for the project. Project scope would exclude: user training, implementation of system in live server, implementation of SSL encryption, mobile (or any other screen size than desktop) version of application.

## Strategy

First of all, research on NoSQL and MongoDB would be carried as part of literature and technology review. Findings from this research would be utilized to implement MongoDB as database solution for resume portal system. This research then would be reviewed with help of project supervisor to carry out any additional research required for the project. Data dictionary required for resume portal would be outlined and reviewed with the help for project supervisor. Different modules of project such as user interfaces, client side home page, dashboard, email feature and privacy modules would be developed and will be review by supervisor. Based on the feedbacks, necessary changes would be made in the system. Finally testing and project evaluation would be carried out. All these phases would be based on the agreed system development life cycle. SDLC for this project is described in proposed methodologies section of this document.

## Deliverables

These are the deliverables required to be produced after completion of the resume portal project. Failure to produce these deliveries would be considered as project failure.

1. Deliverable 1: A operational and well tested resume portal system
2. Deliverable 2: Resume portal should include all features/modules stated in the must have section of initial project plan and specification section of this document.
3. Deliverable 4: Comprehensive literature and technical research document on NoSQL and MongoDB
4. Deliverable 4: An overview document of project implementation
5. Deliverable 5: An overview document of product testing including test plan

## Acceptance Criteria

Resume portal project would be accepted as successfully completed when all deliverable are met positively within cost and time constraints.

## Constraints

Resume portal project has identified few constraints. Project must be ready for demonstration before February 12th of 2017 and final product must be delivered before March 5th of 2017. Research and other documents must be delivered before march 12th of 2017. Cost requirement and schedule requirement for the project as well as feasibility study for the project is attached in the appendix section of this document.

# Review of Literature

NoSQL databases has become integral part of software/web companies in modern world and are indispensable requirement for big data. It is common belief that NoSQL offers great scalability and performance which is very important when talking about fields such as stock market analysis, big data analysis, social network data analysis or weather forecast data analysis.

NoSQL is relatively new technology and requires big investment in terms of both time and money to make system run. If the data is not managed properly it can give great headache to developers. Especially when developers are experienced with relational database, moving to NoSQL requires to put extra efforts. This report studies non-relation (MongoDB specifically) database technology approach to make comparisons with relation database in terms of performance, ease of use eco system and other important criteria such as scalability, availability etc. Report would also study advantages, disadvantages and working structures of NoSQL database. Additionally, basic database operation such as inserting document, updating document is also studied.

## Relational Database

Brief study of relational database would allow to understand its benefits and limitations. Studying NoSQL to overcome these limitations can be beneficial. Relational database is standard database solution in computing world. Relational database stores information in two-dimensional structure called tables where two dimensions are columns and rows (Alagic, 1986). Each table (also known as entities) has predefined columns. These columns define the structure of table. Table can only store information based on columns. Each table store data in term of rows as show in figure 9. Each row represents a set of information for columns. This type of database is popular due to their ability to maintain integrity, storage retrieval and structure.

As the name suggest, tables are also known as relation as it consists of similar type of rows. Relational databases allow to establish and maintain relation between tables with defined set of rules. These rules help to mitigate data redundancy. Rules followed by database are also known as integrity rules that enables relational database to contain accessible and correct information. Structured Query Language (SQL) interacts with database for retrieving, storing, updating or deleting information. Relational database’s ability to support SQL offers great ease which is not possible with flat file type databases. Despite relation database being standard database used in modern applications, it has some limitations that has forced developers to look into alternative solutions like NoSQL database. Some of the examples of popular relation databases are oracle database, MySQL and MSSQL.

## Limitations with Relation Database

The whole NoSQL movement is founded on the limitation of relational database. Relational database is not suitable for all types of requirements. Relational database has been core database solution for most organization for few decades. As the need of horizontal scale and volume growing most organizations has started to choose database based on requirement instead of choosing relational database as default selection.

Some of the core limitations of relational database that prevents relational database from being suitable for all type application are as follows:

### Schema Change

While developing an application changes in requirement is quite often. With relational database being developed first using pre-defined schema making changes to database makes immense impact on application itself. It does not mean applications becomes unusable however, application module associated with tables that is changed also requires modification. Adding or removing columns from table that already has data is very complex task. Relational database is simple not designed for supporting regular changes. When there is need of frequent changes in database of application, it is indication that more flexible solution is required.

### Complex Join

When retrieving, information using join from too many different tables, join becomes very complex and it hinders the performance of query. Getting information for too many normalized tables results performance degrades on query response. In example shown in figure 10 (appendix A), in such scenario when there is large number of relationships, joining tables to get information is quite complex task.

### Slow Query Performance

Despite fine tuning the SQL performance, SQL queries might not be good enough to satisfy the requirements. In such cases, relational databases may affect the performance of the application itself. This issue is often tried to be solved by database de-normalization however, this results bad impact on data quality.

## NoSQL misconception

One of the common misconception about NoSQL is that NoSQL stands for “No SQL”. It is misconception that NoSQL is opposite of SQL. But in reality, NoSQL term stands for “Not Only SQL”. NoSQL is alternative to relational data base and offers many functionalities of relational database. Graph type NoSQL database even have concept of relationship. NoSQL provides developers opportunity of scalability, availability and performance.

## Features

### Schema agnostic

Schema of a database is comprehensive description of database structure and all objects inside database. Schema is an integral part of relational databases as it defines all required elements of database. However, NoSQL database are not bind by a schema hence provides much needed flexibility. Any data can be added into database without worrying much about the structure of database. In NoSQL, schema for the database solution is not necessary to be planned before the programming or coding process.

Developer can start store and retrieve from database without worrying about the internal design of database due to its schema less nature Fowler (N.D.). Key benefit of this NoSQL database feature is impact on the length of project is shortened. Traditional relation database required schema redesign if there is need of additional column in existing database. Then the structure of database objects such as table needs to be modified accordingly. Furthermore, if database has existing data, it can give more headaches. However, in NoSQL, developer does not require to go through process of making changes into the internal structure of database itself.

### Nonrelational

In relation database, tables are required to establish relation with related data tables. In NoSQL database, relations are not required and data are stored or retrieved from database as aggregate data, writes Fowler (2015). Here aggregate data gives meaning of single record containing data of all related tables would contain in relational database. For example, when a person adds multiple email addresses, instead of adding multiple rows in tables, all emails can be added inside person in single file. This gives advantage of getting required information from single file instead of looking into multiple tables through join queries.

### Highly distributable

NoSQL database exhibits highly distributable feature. Highly distributable describes ability to use multiple machines to provide database service instead of using single machine. This is very complicated process but if done right it give great benefits. There can be scenario when a single machine might not be able to handle storage or processing of large dataset. In such cases, distributed system becomes essential. With database distribution, system uses multiple smaller machines instead of one large machine.

## Types/NoSQL Classification

NoSQL database is categorized into four categories as follows:

## Key-Value Store

The simplest category of NoSQL is key-value store. It offers a key and its value, developer can read, write or delete for the key. In simple words, each single value is linked and accessed via their corresponding key as shown in figure 20. Key-value uses hash table to create pair between unique id (key) and pointed value. In an enormously large database, hash tables help to find both simple or complex values. Unlike general concept of database, some key-value stores do not allow to query values but allows keys. Here, values in key-value store can be anything from a song file to string value. As these types of NoSQL are very simple, it offers great speed performance (Stouffer, 2015). However, key-value store is not suitable for using them as main database in applications. Common uses of key-value store are in online shopping carts or creating user sessions where there is defined expire time set. Another use of this type store is in application cache.

Some of the key example of key-value database are Riak and memcached. Data in memcached is not tenacious however in Riak, data are persistent. This means memcached data is lost when node is closed but in Riak, data is not lost.

### **Column family stores**

Column Family database has similar but not exact concept of tabular database. In this database, data is stored in columns and is related with a row key. Column families are set of related data just like in tabular database where rows are set or related data. However, in column-family database each row is not necessary to have same structure of columns Any column can be removed or added for a row based on required. This database has feature called super column which is basically a container for group of columns. Some of the famous and successful example of column family store are Cassandra, HBase, Hyper table and Dynamo DB etc. These types of databases are popular for their fast and easier scalability.

## Graph Store

Graph store has similar feature to relation database such as entities and relationship. However, in graph store entities are called nodes and relationships are called edge (Fowler and Sadalage, 2012). Just like in relational database, edge consists of directional property. Each entity can consist of properties similar to properties in tables of relational database. Graph store allows to store data then taken in dissimilar methods based on relation between entities. When adding relation in a relational database which already consists of data, it requires lots of data movements and structural changes. Graph databases does not have to face these problems.

There are different types of edges in graph store database. This allows to describe relation between notes and mean while also describe path, linked lists or category. There is no limitation of how my types and number of relation nodes in graph database have, hence are described in same database. Similar to relational database, relations (edges) in graph store can itself have properties. For example, in social network database, relation can have properties like sharable contents, friendship start date etc. The core feature of graph store database is edges and ability to add relationships with ease. Neo4J, OrientDB or Infinite Graph are some of the examples of graph store databases.

## Document Database

Finally, document databases are fourth and final category of NoSQL database. These databases are very popular type of database due to their easy implementation. According to Tiwari (2011), people often mistakes document based NoSQL database with document management or content management system. However, the fundamental feature of document database is document itself. Database uses documents of JSON, BSON or XML types to store and retrieve information. Document stores information in hierarchical tree structure. It has similar concept to key-value database but unlike in key-value, in document database values itself can be queried. These databases offer great flexibility as all document type can store same data without many problems and store data in similar manner though they are not exactly same. Application with requirement of storying large but varying data usually choose document type as database solution.

Document database offers great query capability. Usually data are stored in deformalized form, hence retrieving information from document type is easier than from complex joins in relational database. This offers great performance enhancements.

Any data from tabular database can be converted and stored in document database. Data can be stored in either normalized or de-normalized form. In example above, JSON document type is used as document database which is used for storing user info and address. Now think of hundreds of tables and need of retrieving information, this process can take lots of time and is very complex to query. However, as in example above if data is stored in document database, querying data is much faster and simpler.

## Why Document Type

Unlike in traditional relation database where schema for database is a fixed structure and all rows in tables must follow same schema, document oriented database is schema free. This means data in document database can have different model. For example, one customer can have image name field and another customer may have PAN number field. Document type database does not hold space for particular attributes and can be added or removed with ease. IOT applications, mobile, desktop and web all can work together thanks to flexible nature of document database.

## MongoDB

Companies of all sizes and for wide variation of applications uses MongoDB as their core database solution. MongoDB allows to change schema quickly as the size of application grows due to its agile nature. This database is open source, which is its biggest advantage over other NoSQL database however it provides great performance, scalability and availability while still having functionalities from traditional databases of being a fully quarriable language and giving consistency.

Like other document based database, MongoDB offers dynamic schema which allows database to store information with different structure. Released as open source in 2009 mongo stores data in JSON like structure. MongoDB has since grown into becoming one of the major database solution that is being utilized by some of the biggest web companies including SOURCEFOURGE, new your times and eBay. It is cross platform solution that can be installed in MAC, Linux and Windows.

### Why MongoDB

With MongoDB in hand free of cost, there are limitless possibilities starting from high quality cheap and small products to enterprise level large products. Here are some of the reasons why MongoDB was chosen for this research as well as possible reasons for selecting MongoDB for a product their database solution.

         **Rapid** **Development**

Unlike rigid structure of relation database, MongoDB offers dynamic schema, flexible data structure and drivers for development which are well documented. This makes development phase very rapid. Even if there are requirement changes in product, adapting to changes are quite easier than with the relation database. Another important aspect of MongoDB that allows rapid development is in mongo DB there is no need to worry about data validation. Developer can store or retrieve any information without worrying about schema of database.

         **Dynamic Schema**

When MongoDB database is used for a project, there is no need for designing database in advance. In past or even in present when relation database is used as database solution first database is planned and developed then only work on actual product starts. Now that is not the case of MongoDB, database case can be generated dynamically based on the information to be stored. When there is need of additional attribute to be added on table or an attribute is to be removed from table in relational database, production work needs to be stopped for a while. But in mongo this process is not even required due to dynamic schema.

         **Scalability and Availability**

Being subset of NoSQL database, MongoDB inherits benefits of scalability feature. This database can be distributed across data centers. This offers great scalability and availability. With the volume of data increases, additional datacenters can be incorporated. To provide availability, each datacenter just needs to mirror domain data center.

         **Lower Cost**

First of all, as already mentioned in this report above, MongoDB is a free open source database solution. Then price is again reduced as Mongo can run on commodity servers.

* **Query Support**

With MongoDB, rich applicational can be built due to its support for large number of query types. MongoDB supports but not limited to range, Boolean operators, aggregation or key-values types of queries. Query can be performed based on field of the document. MongoDB also supports operators such as OR, AND, NOT while querying information. With these type of query supports, mongo can help in rapid development of application.

## MongoDB Terminology and Concepts

MongoDB stores data in document which consists of value and fields. Value can be another document, array, number, string etc. Documents of MongoDB are in BSON format which is Binary JSON. Just like tables in relation database, document has collections. As this report focuses on study of MongoDB and comparison with relational database here are few MongoDB terms and their SQL counterparts.

         **Collection**

In MongoDB, collection is similar to what tables are in relational database. Documents are stored in collection. Unlike in relational database collection are created by mongo itself while storing document if it is not available.

         **BSON Document (document)**

Just like in relational database data are stored as row, in Mongo data are stored as BSON document.

         **Field**

Fields in MongoDB are equivalent of tables in relational database. A value is associated with each field.

         **Primary key**

Primary key in MongoDB is set automatically in \_id field. Each BSON document consists of one \_id field that stores unique BSON object and acts as primary key.

* **Auto-Sharding**

Sharding is process of scaling out database on low cos using commodity devices. Shards are physical partitions that are used for distribution of data. If sharding is done efficiently it allows MongoDB to use multiple servers to balance data and load pressures on RAM and disk I/O. MongoDB balances the information in clusters automatically.

MongoDB has built in automatic sharding. Hence data distribution and horizontal scaling can be achieved without complexities. No additional software are required to manage data distribution process. Database sharding are categorized into three categories based on how they work.

1. Hash Sharding

Documents distribution are done based on the MD5 of shard key values.

1. Range Sharding

Document distribution are done based on the shard key values. Closer documents has high chances of being sharded into same shard.

1. Zone Sharding

Zone sharding allows database administrators to plan and implement data placement rules. With these rules, administrator can control how data are placed on shards. MongoDB performs migrations automatically to new zone if administrator makes changes to rules.

* **Pluggable Storage Architecture**

Many organizations that develops software/applications requires to run multiple applications at once. Hence, they need to satisfy application requirements, resources and deployment plans. Multiple application’s needs can be satisfied with single database with pluggable storage architecture instead of running multiple databases. With pluggable technology, same MongoDB data model, security, query languages can be utilized of diverse applications.

* **Compression**

MongoDB is shipped with compression configuration that can reduce storage space up to eighty percent. Additional benefit of compression is network efficiency can be improved with less bit transfer. This provides performance improvement as less network bandwidth is required.

* **Transactional Model**

In MongoDB while a document is updated, it is completely isolated. This is due to ACID property of MongoDB. If any error occurs during update process, whole operation can roll back to its initial stage.

* **Security**

MongoDB’s enterprise edition offers multiple security features to secure database from ever growing security risks. Database authentication can be integrated with window active directory. This would allow administrator to keep database server in secured domain network. MongoDB offers authorization based on roles. To keep track of all actions performed on database by different users, MongoDB provides audit logs. Administrators can study logs to find security concerns and address them. And finally, information in MongoDB can be encrypted. Only users with required roles can access that information.

## Comparison between Relational and NoSQL

The fundamental purpose of the research is to understand NoSQL technology and compare them with traditional relational database. This report first researches on various literatures to present comparative analysis on NoSQL and relational database. Furthermore, practical comparisons are performed to carry out more in-depth study.

### Application Development

Database solution for any application is core requirement for that application. As NoSQL and relational database has fundamental differences on how they are designed and implemented, it is not surprising that development flow of application would vary based on the database selection. This report compares basic application development approach for both database selection.

**Relation Approach**

Application development using traditional relational database solution usually involves planning of database first before actual programming of database and application itself. Software planner and designers first identifies the actors for the application. Both internal and external actors for the software are identified and documented. This allows to draw the basic scope of software to be developed. Next step after defining actors for the application as to identify models for actors. In this phase model and actors are mapped.

Once model and actors are mapped, work on database begins. This is crucial stage of application development. Based on the software requirement and actor-model mapping entities for database are identified and documented. In this phase, attributes of entries are planned as well such as data type of attributes, constraint requirements etc. After planning entity attributes, their constraints and their datatypes, relation between entities are analyzed and maintained. Only after completing these database related tasks work on real program begins. For the next release of program, whole process is iterated.

Given software development approach in relational database comes with various challenges. When a new team requires to take over the project, they first need to study large documentations about existing database system. They may not like the way database is designed but even if they like to remodel the whole database they won’t as this would take a long time and they end up keeping up the database system they don’t even like in first place. Other issues with this approach includes, it is extremely complex to change database during programming stage and would need to iterate from first phase to make sure compatible changes. These complex natures of change support in relational database would give problem if organization needs to add additional features or need to merge with other organization. As NoSQL database, such as MongoDB has feature like dynamic schema, software development under NoSQL is slightly different.

**NoSQL Approach**

One of the key benefits in NoSQL approach is, engineers don’t need to worry about database structure straight away from the beginning. Similar to relational approach, first actors and models are defined. According to Vaish (2013), then developers can move directly to development phase instead of planning and developing database and its objects such as entities, attributes or relationships. Database in NoSQL approach can be created dynamically during programming stage. MongoDB supports dynamic collection creation which means collection would be created automatically during insert action if there is no collection available already in database. This significantly reduced software development schedule and would support requirement change during any phase of software development with ease compared to relation approach.

### Data Storage Model

Data storage in relational database is quite straight forward. Each table stores their new record in new row. Each column in row describes specific portion of record. For example, in table ‘Student’ record of a student ‘Mike’ is stored in a row. This row would have columns for storing specific values for Mike’s record such as his name, class, roll number, address, email or phone number. Relational database is structured similar to a spreadsheet solution. Complete information about Mike is not stored in single table. Multiple tables are used for storing his account information, exam records, attendance information or library information. These related tables are bound by relationship to maintain cross reference and data quality. Information from multiple tables are gained by JOINING tables. Joining account and student table would allow to retrieve Mike’s account as well as other information from student table such as email, address or name etc.

NoSQL handles data storage differently based on type. As already mentioned earlier in this report, NoSQL has four categories with each storing information showing some differences on how they store records. Key value type store record similar to tables in relational database. However, key value only has two attributes where one store value and other stores key for that value. Each value can be access via their key only. Another NoSQL type column family stores manages records in more similar manner to relational tables. In column family, record is stored in row and has a row key associated with them. But unlike rigid structure of relational database, rows of column family can be dissimilar to each other. Graph type NoSQL also show similar data storage functionalities to relational database. In relational database, relation between tables are maintained via primary and foreign keys and this feature is ditched by Graph.

Finally, document database such as MongoDB entirely changes the way data are stored. It stores data in form of BSON document. Each record is stored in form of new document with unique key associated with it. Comparing with example of how record of student is stored in relational database, all information for ‘Mike’ is stored in single document. This allows to retrieve all information using single simple query instead of writing and executing complex JOIN queries in relation database.

### Schema

Schema for relational database is required to be pre-designed and pre-planned. This means relational database as well defined inflexible schema. Designing schema is very critical phase as it defines what information can database store. It is not possible to add new table in relational database without making change to whole database itself and requires to be offline. In MongoDB, a document type NoSQL is schema less database. This means schema is not something to worry about in MongoDB. Any required field can be added dynamically whenever required. Data structure is not necessarily pre-planned and still any field can be added in document on fly.

### Vendors

It is important to compare how each database type is getting support from their vendors. SQL databases has been standard database solution for long time. Support for SQL database can be easily available from independent sources. More importantly, it should not be forgotten that vendors for SQL database are Microsoft, IBM or oracle are some of the largest IT companies of the world. These vendors are providing admirable supports, update, patches and documentations for their respective database. It is these three vendors due to which SQL server, oracle database and DB2 database, specially first two has been first choice database solution in current market.

NoSQL on the other hand, might require direct support from their vendor to deploy them as it is likely that experts on particular database not accessible in surrounding. As there are limited number of experts compared to relational database not all organization are showing much interest in NoSQL. However, for small scale deployment large community support are available with ease. For example, MongoDB, Cassandra, HBase and Neo4J are some of the most famous NoSQL database according to reports and quite excellent supports are provided buy their vendors.

### Eco System

As already discussed in this document, NoSQL database can use number of commodity servers instead of single expensive server compared to relational counter parts. This gives NoSQL huge advantage when it comes to cost of running database system.

However, it is important to note that database server for NoSQL are quite pricier than that of relational database servers such as MySQL or MS SQL. This might be due do the fact that relational database has been international standard for many years and database engineers for SQL are easier to find compared to NoSQL.

### Normalization in SQL and De-normalization in NoSQL

SQL stores data in normalized form whereas NoSQL prefers de-normalized form. Buckler (2015) suggests, normalization of data offers ability to reduce data redundancy. Data does not require to be repeated and can be referenced using key. This allows data to be updated without much issues. For example, if an employee has role column and referenced with roles table then making changes in roles table can be reflected on employee table. And to change role of employee, just role key needs to be modified. NoSQL also allows data to be stored in normalized form but it is not always practical hence data is stored in de-normalized form. This offers improved query performance however, need of repeating data for each record makes data write slower.

### Data Integrity

In SQL database, data integrity is enforced through rules called integrity rules. This allows to maintain usability of data and data quality and prevent creation of invalid data. Foreign key rule in SQL prohibit developer to enter invalid data (Balter, 2005). Furthermore, primary key for a table cannot be null. However, these data integrity rules are not available in NoSQL database. Developer can insert any data as there is no foreign key rules to prohibit it. Due to dynamic schema of NoSQL database such as MongoDB each document can store data of different data type. However, each document in MongoDB as unique BSON id.

### Transaction Model

In relational database, it has ability to put updates in transaction. This means when two or more tables are kept in transaction, either all of them gets update at once or none of them gets updated. Failure to make update in single table will result roll back whole process. This will ensure validity of data. For example, when kept inside transaction, both sales and storage logic are executed at once. Either both sales and changes to storage table is made successfully or neither of them is executed.

Plugge et al. (2010) suggests that MongoDB might not be ideal database solution for account related application due to absence of transaction. MongoDB however does offer some level of transaction when updating on single document. But multiple document transaction is absent in MongoDB as well. Multi document transaction can be achieved by manual coding in application.

# Review of Technologies

## Data Access [CRUD]

In any database system, creation, reading, modification and delete are fundamental operations. Relational database performs these operations through SQL while NoSQL has their own queries. SQL and NoSQL has their own set of queries for these functionalities that are distinctively different to each other. This report compares how SQL and NoSQL performs these basic operations while also taking special functionalities such as aggregation into consideration.

|  |  |  |
| --- | --- | --- |
| Operation | SQL Term/Syntax/Concept | NoSQL Syntax/Concept |
| Inset | INSERT INTO student (  name, subject, Roll  )  VALUES (  'Atut Gorkhali',  'Web development',  1  ); | db.student.insert({  name: "Atut Gorkhali",  subject: "Web development",  roll: "1"  }); |
| Update | UPDATE student  SET subject = ‘computing’  WHERE roll = 1 | db.student.update(  { roll: ‘1’ },  { $set: { subject: ‘computing’ } }  ); |
| Read | Select \* from student | db.student.find() |
| Drop table/collection | Drop table student | db.student.drop() |
| Delete | Delete from student where roll=1 | db.student.remove( { roll: "1" } ) |
| Count | Select count(roll) from student where subject=’computing’ | db.student.count({  "subject": "computing"  }); |
| Sum aggregation | Sum() | $sum |
| Manage order | ORDER BY | $sort |

Above table shows syntaxes for performing some fundamental database operations. In both SQL and NoSQL (taken mongo dB for this example) has different syntaxes and one may prefer to another based on their knowledge and experience on particular technology. As documentations for both technologies are available with ease, finding solutions for problem related to CRUD and other basic operations are not hard. However, due to limitation of NoSQL experts in local market, there might be some problem to find solution for complex queries.

## MongoDB in web application development

### MongoDB C# driver

To work with MongoDB in C# programming, MongoDB driver is required to be downloaded. To make development easier Robomongo application is needed to be installed. Robomongo is free GUI tool for MongoDB database management. **After installation of MongoDB, robomongo can be installed and new connection requires to be created using portal 27017 and having address as localhost. After setting up MongoDB and robomongo, drivers for MongoDB can be downloaded from Nuget store.**

## Practical Comparison

During technical review, a practical comparison between implementation of MongoDB with C# and SQL database with C# is performed. To do this coding for saving a resume on both SQL and MongoDB is written below.

Two different code segments for saving a resume on SQL and MongoDB are provided in **Appendix F**. As MongoDB store information in de-normalized aggregated form, whole resume is inserted as document at once. However, SQL stores information in normalized for to mitigate data redundancy. Hence, information about education, job, personal details, skills are stored in different tables. This results a lengthy code for saving a resume which required far less coding.

# Proposed Methodologies

## Project Life Cycle Model

It is very important to choose correct project development model for resume portal. As it provides framework for how project should be managed. Resume portal project would follow modified version of water fall methodology for application development. It would offer project more flexibility due to its ability to overlap phases on as required (IJCSI, 2010). Key phases of project are based on water fall model starting from requirement analysis and ending at deployment/maintenance. However, project is adapted to support any requirement changes based on the feedbacks from project supervisor. Here are key phases of the project:

1. Requirement Analysis/ Research

Research about CV database requirements, use of MongoDB as database and need of supporting tools and technologies were carried out in this phase of the project.

1. Design

Various system design tools such as data flow diagram, use case diagrams, flow chart, sequence diagram and context diagram is designed during this phase.

1. Development

After research and system design, prepared documents are utilized to develop (program) actual system.

1. Testing

Testing of the developed resume portal system is carried out after development phase is completed. Bugs are record as test log. This test log is later utilized during debugging session.

During or at end of each phase, feedback from supervisor is taken and appropriate changes are made. Adapting to changes are made possible by use of modified version of waterfall model.

## Model View Controller (MVC) Pattern

To develop the resume portal product, MVC pattern approach would be implemented. By this mean, three major concerns of application. In MVC pattern, M stands of model and represents data interaction. V in pattern stands of View and represents user interface for the data (model). And finally, C stands for controller which is basically a platform to control the flow of data and update of views. Pattern is implemented in order of M ,C then V rather than the name of pattern which is M, V and then C (Chiaretta and Nayyeri, 2009). For the resume portal, MVC pattern would be implemented in following method:

1. Model for resume, session data, User info, home page data and change password request will be developed first
2. Business logic for different modules of product will be developed
3. Controlled will be programmed to control the flow of data
4. Views for controllers based on their actions would be added to display data accordingly

With the help of MVC pattern, codes would be more managed, reusable and clean. This would help to make necessary changes to project after feedbacks from supervisor or debug the system after testing session.

# Product Design

Now the requirements for project has been identified and proposed scopes and methodologies for project has been described, various system design tools such as UML is utilized to prepare design for resume portal. Diagrams such as UML, context and sequence diagram and data dictionary for product is proposed below.

## Use Case Diagram

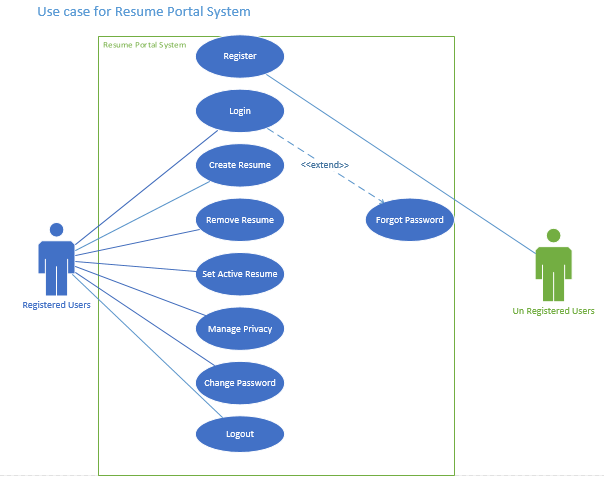


Figure 1Use case diagram for resume portal

Product would have two types of user, registered users and un-registered users. Un-registered users would be able to register into the system while registered users would login, logout, create and view resume, find resume change password or image and set or remove resume from their profile.

## Flow Chart

Figure below describes the flowchart for the proposed resume portal.

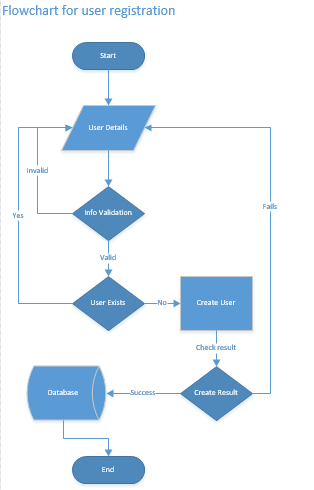


Figure 2 Flowchart for resume portal (User registration)

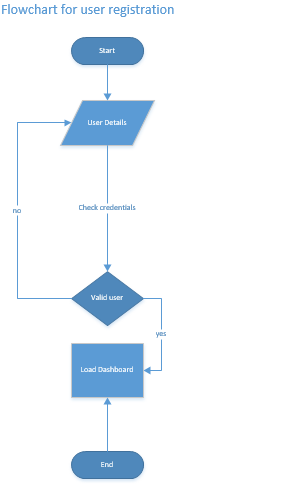


Figure 3 Flow chart for resume portal (User Login)

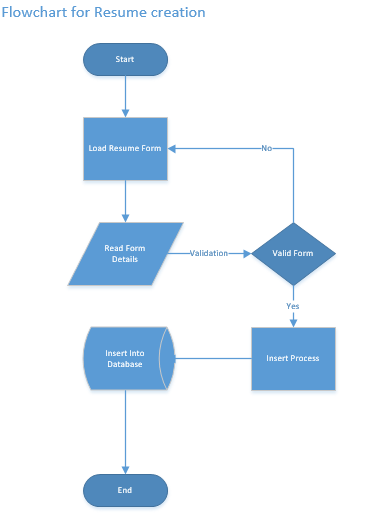


Figure 4 Flowchart for resume portal (resume creation)

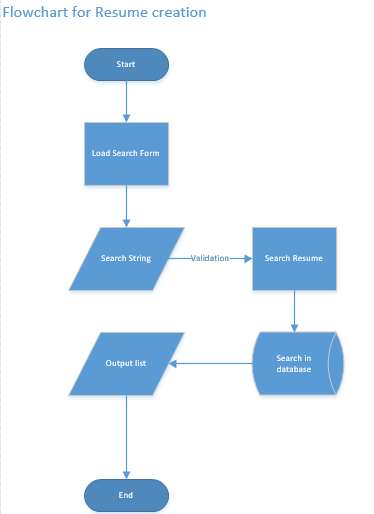


Figure 5 Flowchart for resume portal (find resume)

## Data Flow Diagram

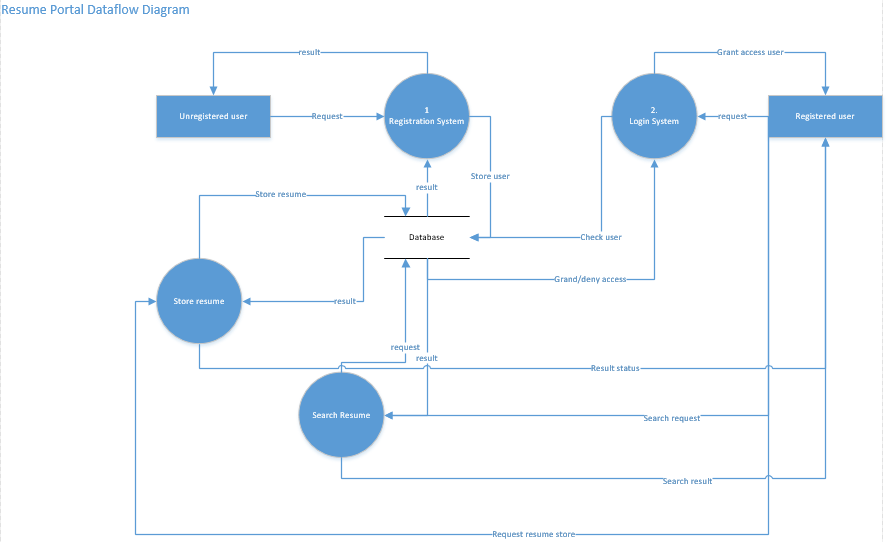


Figure 6 Data Flow Diagram for resume portal

## Sequence Diagram

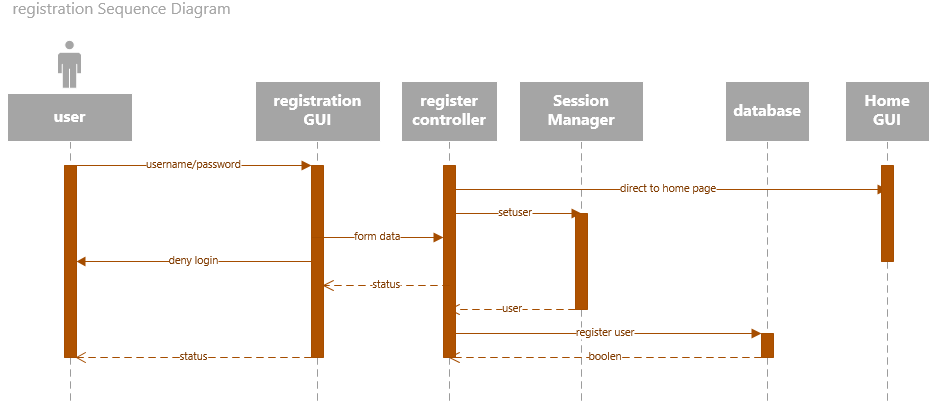


Figure 7Sequence Diagram for resume portal (registration)

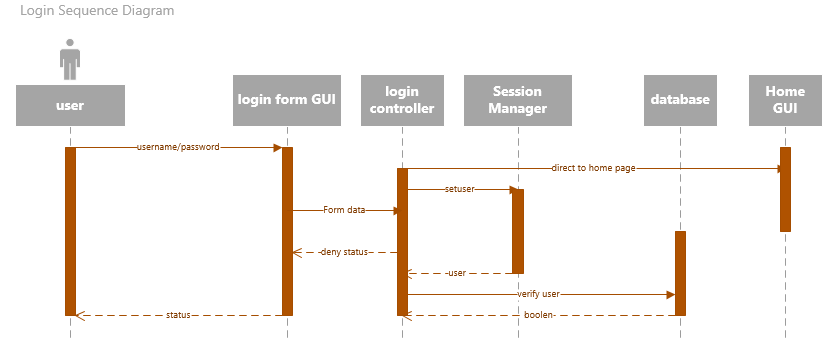


Figure 8Sequece Diagram for resume portal (login)

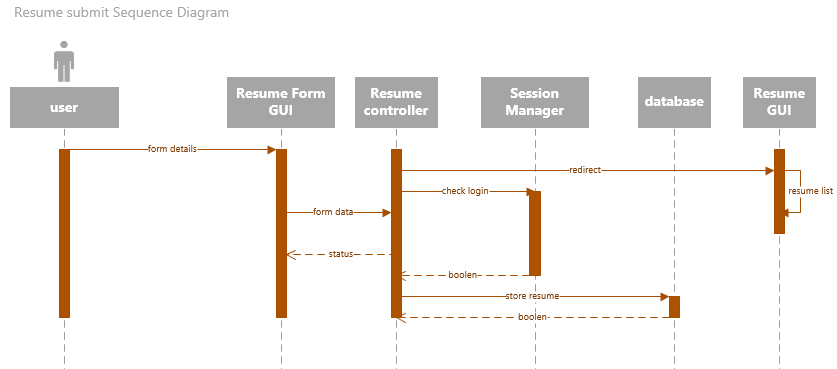


Figure 9 Sequence Diagram for resume portal (resume form submit)

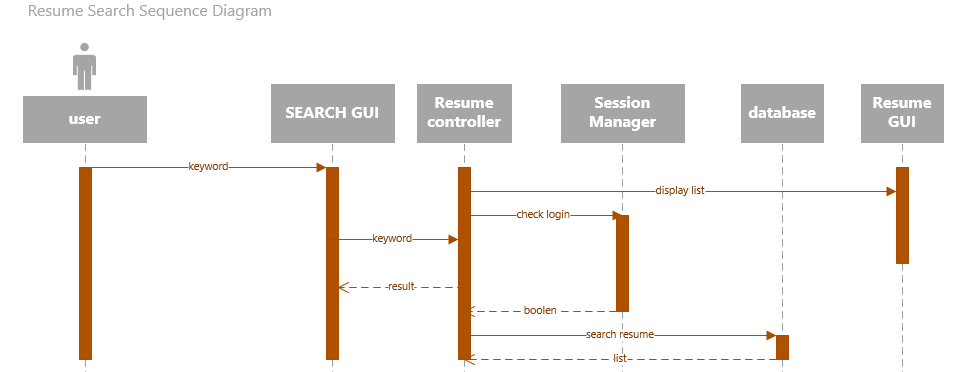


Figure 10 Sequence Diagram for Resume portal (resume search)

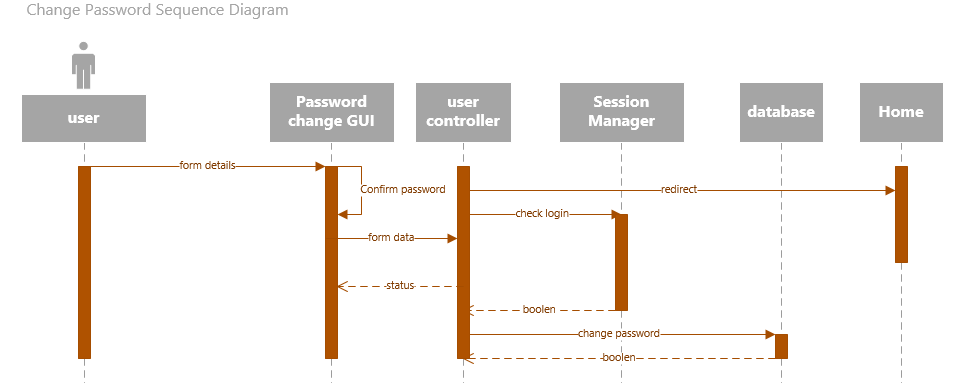


Figure 11 Sequence Diagram for resume portal (change password)

## Context Diagram

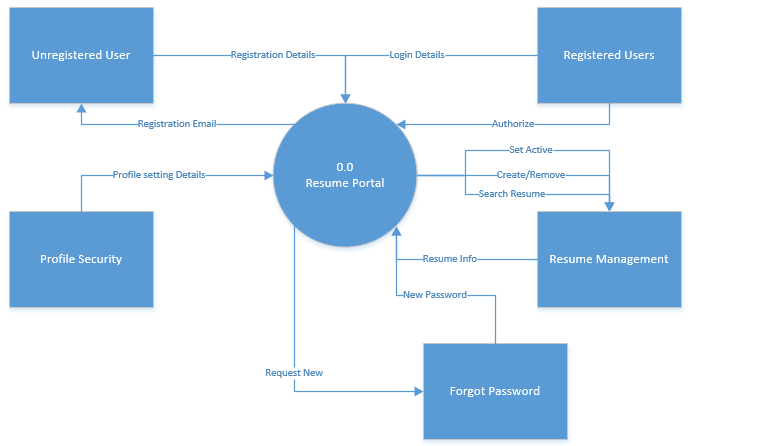


Figure 12 Context Diagram for resume portal

## Data Dictionary

Corfield (2007) states resume should cover information about personal details, personal profile, key skills, Employment history, Education etc. Proposed resume portal would store above stated details while being clear and concise. Data dictionary for resume portal is describe below.

|  |  |
| --- | --- |
| Field Name | Requirement Description |
| Username | stores username |
| Password | stores password in md5 form |
| User Image | stores profile image location |
| Profile category | Stores profile category such as engineer, teacher, business etc. |
| User id | Unique id for each user |
| Resume id | Unique id for each resume |
| keywords | Keywords for resume |
| objects | Resume objective |
| Education id | Unique id for each education |
| Education description | Describe education |
| Education from date | date when education started |
| Education to date | Date when education finished |
| Is education Active | Specify if education is active |
| Job id | Unique id for each Job |
| Job description | Describe Job |
| Job from date | date when Job started |
| Job to date | Date when Job finished |
| Resume image | Location of resume image |
| Set active | Set resume as active |
| Language id | Unique id for language |
| Language Name | Name of language |
| Language level | Level of language |
| Skill id | Unique id for Skill |
| Skill Name | Name of Skill |
| Skill level | Level of Skill |

### Functional Requirements

Following are the functional requirements design for the proposed resume portal project: -

* User Registration System: Any visitor of project website should be able to register into the system freely.
* User Login System: Any user with valid username and password should be able to login into the system.
* Resumes Summary in Home Page: Project’s Home page should display summarize information about types of resumes, total users and total resumes
* Profile Modification: Any logged in user should be able to change their profile picture, name or public link name.
* Password change: Any logged in user should be able to change their password.
* Create resume: Any logged in user should be able to add new resume
* Resume search: Any logged in user should be able to search existing resumes based on education, location or keyword etc.
* Notification System: Users should receive email notification for password reset and new user registration.

These design documents would help developers to develop resume portal. Based on the planned data dictionary, models are developed. Use cases, data flow and flow chart helps developed to plan controllers and views for the portal. After reviewing these designs project moves to implementations and testing phase.

# Product Implementation and Testing

## Implementation

### MongoDB Implementation

Implementation of MongoDB is quite straight forward. MongoDB for windows is downloaded first from official website. Downloaded file is extracted into hard disk first. Then a config file, log file and data folder is created. Config file consist information about location of log file and data directory. Now using config file MongoDB is installed into the system and a service is created as well from command prompt. Following scripts is executed in compand prompt.

mongod -f "C:\Server\mongodb\config\mongodb.conf" --install --serviceName mdb27017 --serviceDisplayName "MongoDB Server Instance 27017" --serviceDescription "MongoDB Server Instance running on 27017"

### MVC Pattern

As mentioned in methodologies section, software development of this project follows MVC pattern. Implementation of pattern is explained below.

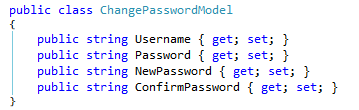


Figure 13 Model in MVC

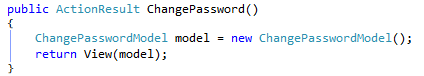


Figure 14 Controller in MVC



Figure 15 View in MVC

### Service Layer

Service layer sits behind controller and helps to keep controller cleaner. Following code for saving or updating a resume is kept in service layer class to keep resume controller cleaner and manageable.

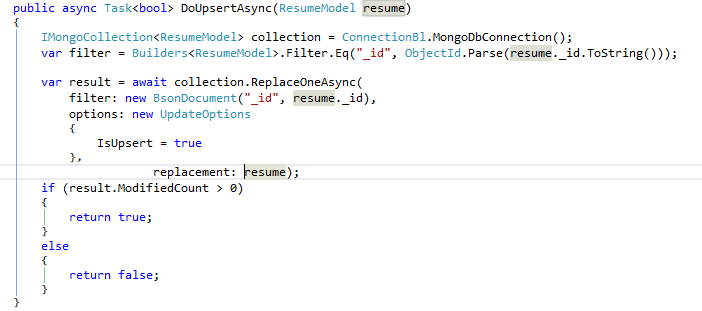


Figure 16 business logic for insert/update resume

### Sending Email

To implement email notification and email password recovery functionality, Gmail’s SMTP is utilized as shown in figure below.



### Publishing

Once development of resume portal completed, it was published in local IIS server. To do so, in visual studio, under build section, publish meu was selected. And then web application was published. In IIS manager (INETMGR in run menu) these published files were used for creating a new website. For testing purpose, [www.portal.com](http://www.portal.com) domain was given to this test application. Resume portal was accessible from browser via [www.portal.com](http://www.portal.com).

## Testing

Software testing is one of fundamental process for building quality application. Application is examined with intention to find any deficiencies in program such as software bug. This helps to developer quality application with bug as less as possible. After development of resume portal (completion of programming portion), comprehensive testing of the portal was conducted and reported in test document. Validation testing, unit testing and compatibility testing of developed resume portal was conducted and reported in terms of test logs. Resume portal test emphasizes on functional requirements, response to inputs and ability to run in different environments.

### 6.2.1. Unit Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N.** | **Date** | **Title** | **Description** | **Expected Result** | **Actual Result** |
| 1. | 3/1/2017 | User registration | Submitting registration form with required information to register new user | New user should be registered then logged into the system | New user was created and user was logged into the system |
| 2. | 3/1/2017 | User login | Use of correct username and password should log user into dashboard | Successful login | Successful login |
| 3. | 3/1/2017 | Create resume | Submitting resume form with required fields filled should store resume | Resume should be created | Resume created |
| 4. | 3/1/2017 | View resume | Clicking on view icon in resume list should show preview of resume | Should open new page with resume | Resume page opened |
| 5. | 3/2/2017 | Modify privacy | Should modify privacy | Update privacy | Privacy updated |
| 6. | 3/2/2017 | Modify profile | Should modify profile | Update profile | Profile updated |
| 7. | 3/2/2017 | Print | Clicking on print button should show print wizard | Open print wizard | Opened print wizard |
| 8. | 3/3/2017 | Change password | Clicking on change password with correct old password and matching new password should update the password | Update password | Password updated |
| 9. | 3/3/2017 | Send email | Reset password request should send password to registered email | Should send password via mail | Received password in recipient email |
| 10. | 3/4/2017 | Password Encryption | Passwords should be saved in encrypted form | Password should be saved encrypted | Password saved in encrypted form |

### 6.2.2. Compatibility Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N.** | **Date** | **Title** | **Description** | **Expected Result** | **Actual Result** |
| 1. | 3/1/2017 | Chrome Browser test | Test chrome browser | Completely load and run application | Successfully loaded and application ran |
| 2. | 3/1/2017 | Mozilla Browser Test | Test Mozilla browser | Completely load and run application | Successfully loaded and application ran |
| 3. | 3/1/2017 | Opera Browser Test | Test opera browser | Completely load and run application | Successfully loaded and application ran |
| 4. | 3/1/2017 | Mobile Browser (Samsung S III) | Test Mobile browser | Completely load and run application | Application run but failed to load some icons |
| 5. | 3/1/2017 | Tab Browser  (Samsung Tab 7.0) | Test TAB browser | Completely load and run application | Application run but failed to load some icons |

### 6.2.3. Validation Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N.** | **Date** | **Title** | **Description** | **Expected Result** | **Actual Result** |
| 1. | 3/1/2017 | Registration Form required field Validation | Registration should require all fields | Show error message when blank field submitted | Showed error |
| 2. | 3/1/2017 | Registration Form email validation | Only valid email should be allowed | Should throw error message when invalid email submitted | FAILED  Invalid email was accepted |
| 3. | 3/1/2017 | Registration Form password validation | Registration form should require matching confirmation password | Should throw error when different password submitted | Showed error message |

This report has documented various tests performed on developed resume portal. During the tests, some of the issues were identified such as issue with email validation. Such issues are reported and debugged during maintenance of product.

# Product and Project Evaluation

## Product Evaluation

After successful completion of development and testing of resume portal, it is crucial to evaluate the potentials and limitations of product for future improvements.

### Strengths

#### Responsive and User friendly UI

Resume portal has implemented very simple and user friendly user interface. This offers users ease while browsing through the system. Responsiveness nature of user interface allows web application to run in device or monitor of different sizes. This responsive UI is achieved via use of bootstrap framework. Menus for different interactions are well placed and named correctly which makes UI look clearer and well maintained. Home page of portal offers insight of internal system while also providing information about portal’s user data and process of finding or creating a resume. This helps to attract new visitors to register into the system.

#### MVC and services layer

One of the key strength of developed resume portal is use of MVC pattern and service layer integration during development phase. With MVC, different concerns of web application are separated. This means UI section, data sections and controller for managing flow of data in UI is handled separately. Furthermore, service layer has been implemented. In service layer, core business logics are written for controllers. Use of this MVC and service layer in current project has helped to improve maintainability and testing of project. With this architecture, code cleaning, debugging or addition of more features can be achieved more easily.

#### Email Notification

Resume portal system has successfully implemented email system. This feature of product has been fully tested and has generated successful result. Gmail’s SMPT service (smpt.gmail.com) has been utilized to implement resume portal’s email service for free of cost as Gmail offers this SMTP for free of cost. With help of this service, users can get notification for their registration. Additionally, users can request for their password via forgot password. Users will receive their credential details through email. This would help to keep old users while also welcoming new users.

### Limitations

Through review of developed resume portal after feedbacks from different users’, various limitations in system has been identified. Evaluations of these limitations would help developer to address them in future releases of product. These findings are also based on the thorough review of test logs from test phase and system design documents.

#### Security

Security is one of key requirement for any web application. Resume portal has implemented some of the basic security measurements such as user authentications and password encryptions. However, with advancements in technology, these security measurements cannot be enough for the security of a system that has personal information about so many individuals. These measurements can easily be exploited with help of bots. Hence security should be kept in priority which current product has failed to do. Additional alternative security solutions should be implemented to make system more robust and secure.

#### ability to modify resume

This resume portal exhibits a basic functionality fault. User has no ability to modify the resume once it has been created. This means to modify a resume he should delete original resume then create a new one. This is a critical design deficiency. Developer must add necessary modules to allow users to make changes to their old resume instead of require to create new ones each time.

### Recommendations

#### Captcha

CAPTCHA is a free tool from CAPTCHA.net which enables websites to prevent bot attacks by offering users to solve some problem. These problems can easily be solved by human users but bot cannot pass through these tests. Implementations of captcha in registration form and login form (after multiple failed try) can prevent website from bot trying to create fake accounts or from getting access into system.

#### 2 Factor Verification (2FA)

As already discussed, security for the resume portal needs to be improved given the fact that individuals would store personal information. Security of product would improve immensely with help of two factor verification which is also known as two level authentications. 2 factor verification is basically an authentication process where requires to provide two security factors such as password, secret code from SMS, email verification or biometric etc. This security enhancement helps to strengthen the overall security of website as any user would require to pass through two authentication level.

## Project Evaluation

A critical product evaluation as allowed to understand strength and limitations of developed resume portal system. Now it’s essential to evaluate whole project to analyze the project, its strengths and limitations for future improvements.

During the project plan phase, project was allocated certain time and was bound by a Gantt chart. This Gantt chart described different phases for project while also stating milestones for each project and their deadlines. Following project schedules based on proposed Gantt chart helped to learn and exhibit professionalism. Aim of the project was to research, plan, design and develop the resume portal and make comparisons on NoSQL database and SQL database. Findings from this comparison research would help to implement MongoDB as database back end for resume portal. During the research phase, comprehensive list of sources (more than twenties) was utilized to review traditional relational database. Then research on NoSQL was concluded including their background, features, architecture, strengths and finally their limitations.

Furthermore, technology useful for project such as MongoDB was studied. A practical research on use of MongoDB with C# was conducted to critically review the development method. This practical research helped to understand practical differences in implementation of MongoDB as database solutions and use of relational DBMS as database solution. It was found that use of MongoDB offers greater ease to developers to during development of system due to their well document driver and schema less nature. Developer can more focus on quality of product and client’s requirement than database structure as any chances made on code can change database structure dynamically.

An improved version of modern waterfall model was utilized for product development for this project. Use of this improved methodology offered ease of simple nature of methodology. To develop resume portal requirement analysis was concluded and documented first. Then documents from analysis phase was utilized to review and validation of progress. Project supervisor provided useful feedback to progress and these feedbacks were implemented for final requirement analysis. Similarly, other phases of project such as planning, designing, development and testing were reviewed by supervisor and necessary changes were made by developer/analyst. This helped to support any requirement changes with ease. Additionally, schema less nature of MongoDB also allowed developer to focus of requirement changes rather than database structure.

### Limitations

#### Performance based comparison

This project partially fails to conduct performance based practical comparisons of NoSQL and SQL database. This was due to given schedule constraint and available technical resources. Any performance based research findings were solely based on literature reviews. However, practical research to prove the cases would had made project more truthful. Practical research conducted during technical review exclusively concentrated on use of both database technology and ease of implementation. This research cannot satisfy the need of practical research to demonstrate performance of different database on resume portal project.

#### Absence of Team (Only two members)

Project for resume portal consists of only two members. One of these members is supervisor who would provide necessary feedbacks for improvements. All phases of project such as research, plan, developer and testing was done by single individual. Completion of all objects for project and achievement of core project aim within given time constrains makes project considered successful. However, use of more manpower with more diverse skill sets would have improved the quality of project. For example, manpower with more expertize on security feature implementation would have allowed to include more security on the product. Having team member with programming skills would have helped to complete development phase in shorter time. This would offer more time for other phases.

# Summary

This project intended to research and develop an online resume portal while also researching NoSQL technology and MongoDB technology. A practical research carried out to compare implementation of both SQL and NoSQL technology as database solution. During the research, it was found that there are four types of NoSQL database. Each database can be suitable for different scenario. NoSQL has become extremely popular due to its horizontal scalability and schema less nature. During the development of resume portal with MongoDB, there were often requirement changes based on feedbacks from supervisor. Each requirement change was easy to handle due to use of modified waterfall model and dynamic schema of MongoDB.

However, practical comparison was limited to ease of use and performance comparison were left out due to technical and time constraints. This research can further extend to carry out practical comparison on performance, cost for bot SQL and NoSQL. Finally, decision to use of MongoDB was fruitful as it helped developer handle changes with ease and complete project within given timeframe. However, with MongoDB product has lost consistency property. And absence of transaction can cause issues. But as resume portal would need to handle large number of data in future, it would enhance query performance as there is no complex joins. And as resume portal is not accounting software, absence of transaction might not cause much problem. If needed, transaction can be achieved via manual coding. For product of resume portal nature, use of MongoDB can be justified however, SQL cannot be completely forgotten due to its ability to guaranty consistency, availability, isolation and durability. NoSQL should not be taken as rival to SQL but rather should be taken as friend that can help development of complex products.

# Bibliography

* Alagic, S. (1986). Relational database technology. 1st ed. New York, N.Y.: Springer.
* Balter, A. (2005). Sams teach yourself SQL Server 2005 Express in 24 hours. 1st ed. Indianapolis, Ind.: Sams.
* Chiaretta, S. and Nayyeri, K. (2009). Beginning ASP.NET MVC 1.0. 1st ed. Indianapolis, IN: Wiley.
* Corfield, R. (2007): How to Make a Great Impression and Get the Job You want. 4th edition India: Replika Press
* Deka, G. and Bakshi, S. (n.d.). Handbook of research on securing cloud-based databases with biometric applications.
* Fowler, A. (2015). NoSQL For Dummies. 1st ed. Somerset: Wiley.
* Fowler, A. (2016). Common Features of NoSQL - dummies. [online] Dummies. Available at: http://www.dummies.com/programming/big-data/common-features-of-nosql/ [Accessed 27 Aug. 2016].
* Fowler, M. and Sadalage, P. (2012). NoSQL distilled. 1st ed. Boston, Mass.: Addison-Wesley.
* Gupta Edward, S. and Sabharwal, N. (2015). Practical MongoDB. Apress, pp.17-19.
* IJCSI (2010). International Journal of Computer Science Issues, 7(5), p.97.
* Jacqueline, J. (2016). Relational vs. non-relational databases: Which one is right for you? [online] Plural sight. Available at: https://www.pluralsight.com/blog/software-development/relational-non-relational-databases [Accessed 27 Aug. 2016].
* Jeon, J., An, M. and Lee, H. 2015. NoSQL Database Modelling for End-of-Life Vehicle Monitoring System. JSW. 10(10), pp.1160-1169.
* Kitchin, R. and McArdle, G. 2016. What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets. Big Data & Society. 3(1)
* Leavitt, N. 2010. Will NoSQL Databases Live Up to Their Promise? Computer. 43(2), pp.12-14.
* McCreary, D. and Kelly, A. 2014. Making sense of NoSQL. Shelter Island, NY: Manning.
* McGee, P. (2003) Writing a CV that Works: How to Develop and Use Your Key Marketing Tool. 2nd Edition edn. United Kingdom: How to Books Ltd.
* Ming Wu, C. 2015. Comparisons Between MongoDB and MS-SQL Databases on the TWC Website. American Journal of Software Engineering and Applications. 4(2), p.35.
* Nepaliya, P. and Gupta, P. 2015. Performance Analysis of NoSQL Databases. International Journal of Computer Applications. 127(12), pp.36-39.
* Patel, J. 2016. Operational NoSQL Systems: What's New and What's Next? Computer. 49(4), pp.23-30.
* Plugge, E., Membrey, P., Hawkins, T. and Hows, D. (2010). The Definitive Guide to MongoDB. 1st ed. Berkeley, CA: Eelco Plugge, Tim Hawkins, Peter Membrey, p.4.
* Stonebraker, M. 2010. SQL databases v. NoSQL databases. Communications of the ACM. 53(4), p.10.
* Stouffer, J. (2015). *Mastering Flask*. 1st ed. Birmingham, UK: Packt Publishing, p.104.
* Tiwari, S. (2011). Professional NoSQL. 1st ed. Indianapolis, Ind.: John Wiley & Sons, Inc.
* Tiwari, S. 2011. Professional NoSQL. Hoboken, N.J.: Wiley.
* Vaish, G. 2013. Getting started with NoSQL. Birmingham: Packt Publishing.

# Appendices

## Appendix A

### Literature Review Images

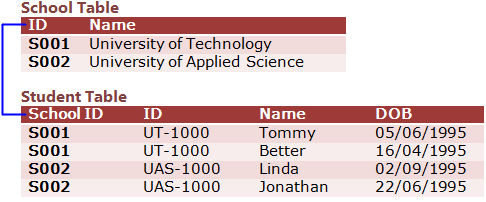


Figure 17Relational Database Structure

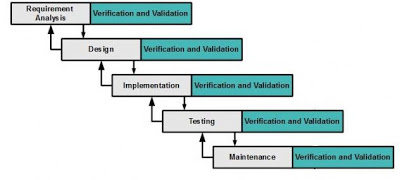


Figure 18 Basic Structure of Modified Waterfall model

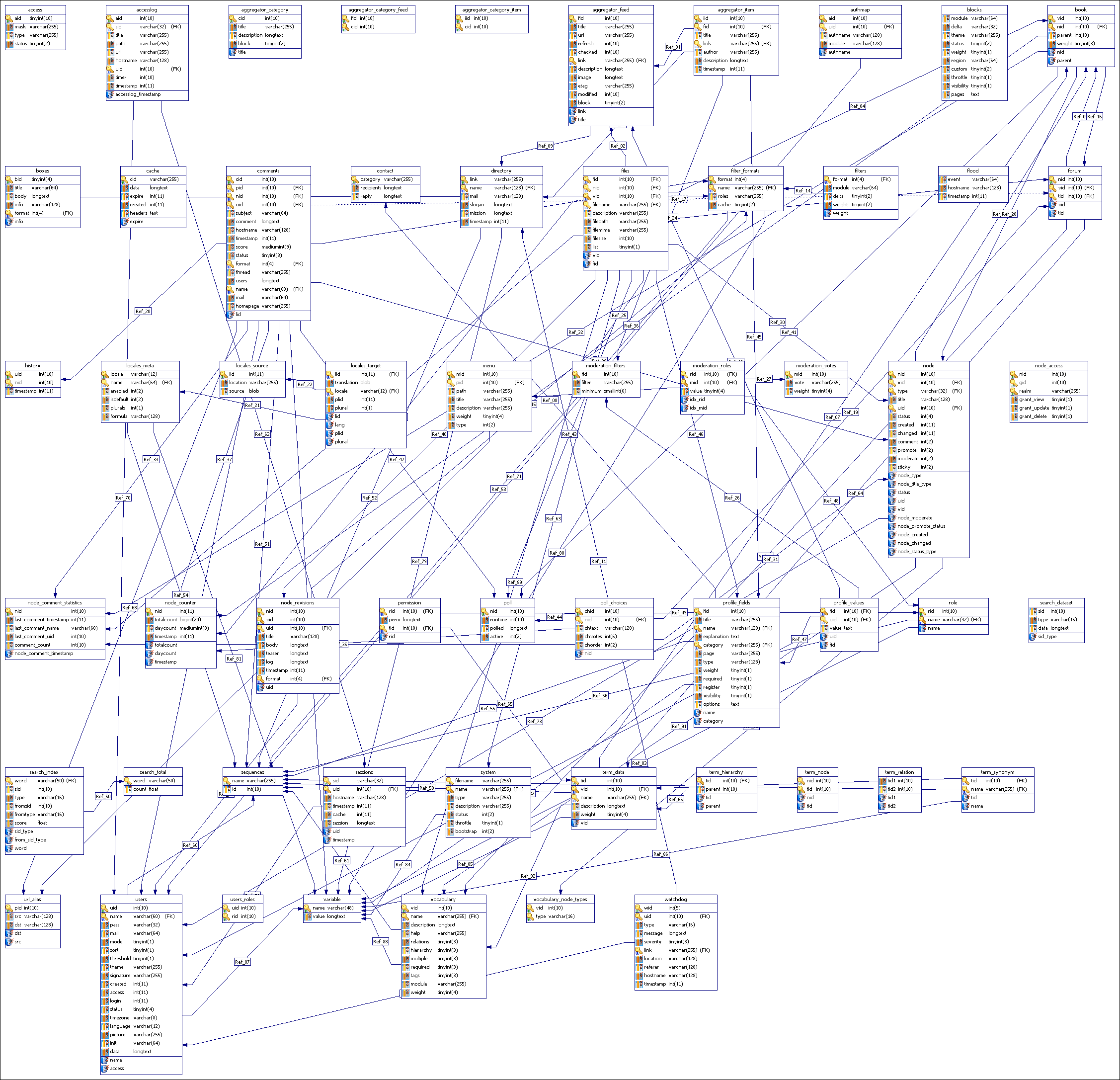


Figure 19 Complex SQL database structure

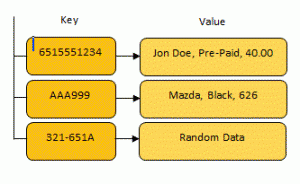


Figure 20 Key Value Database structure

### Technical Review Images

#### C# Driver

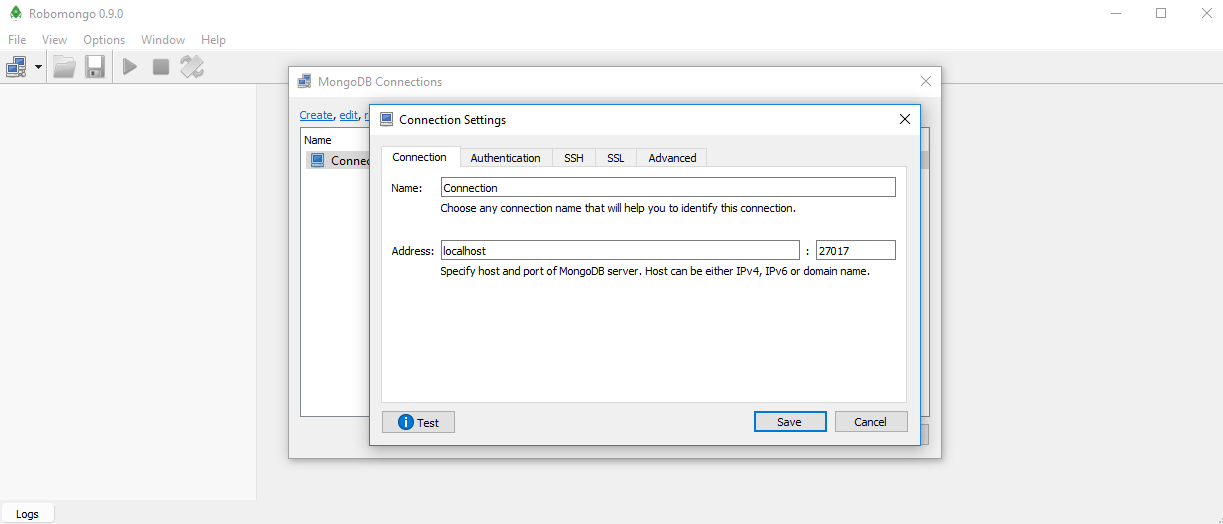
****

Figure 21 Setting up new connection in robomongo

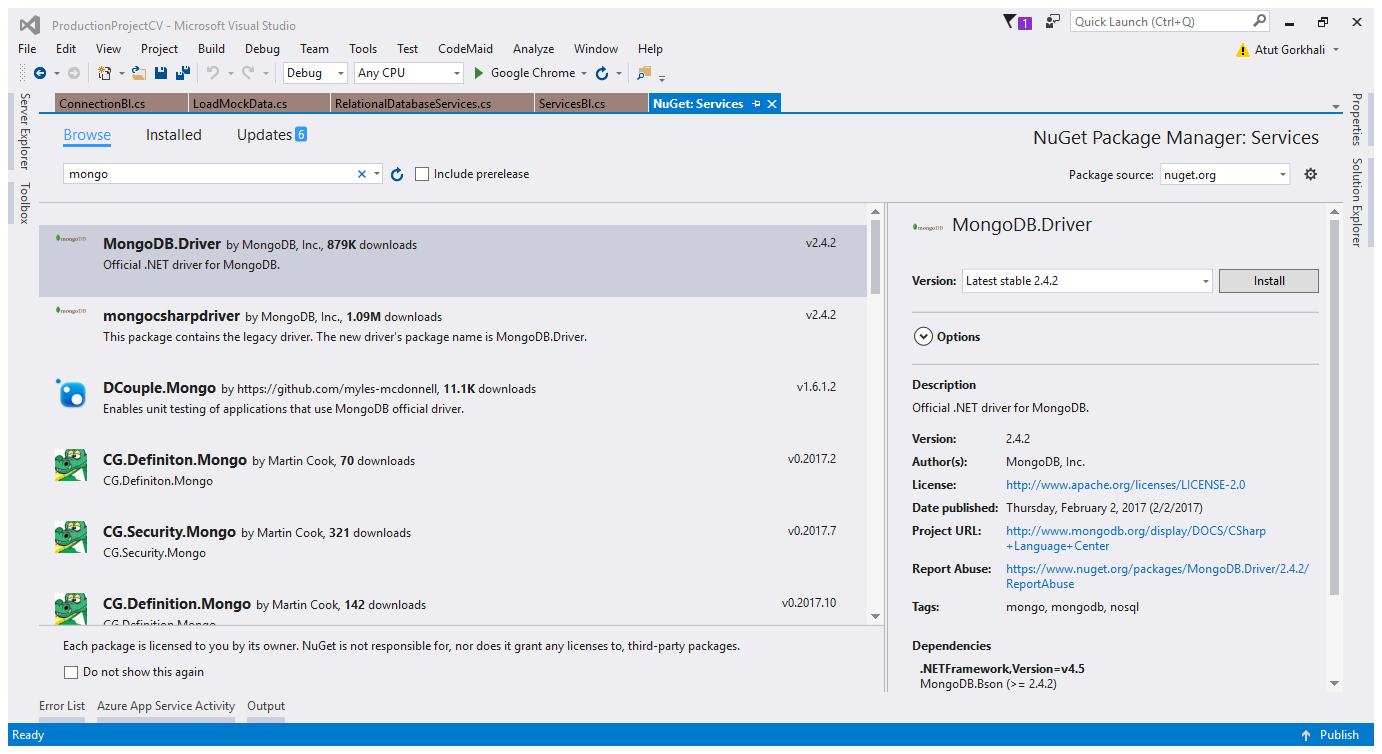


Figure 22Downloading MongoDB in Nuget

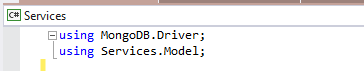


Figure 23using namespace to use MongoDB driver

### Compatibility Test

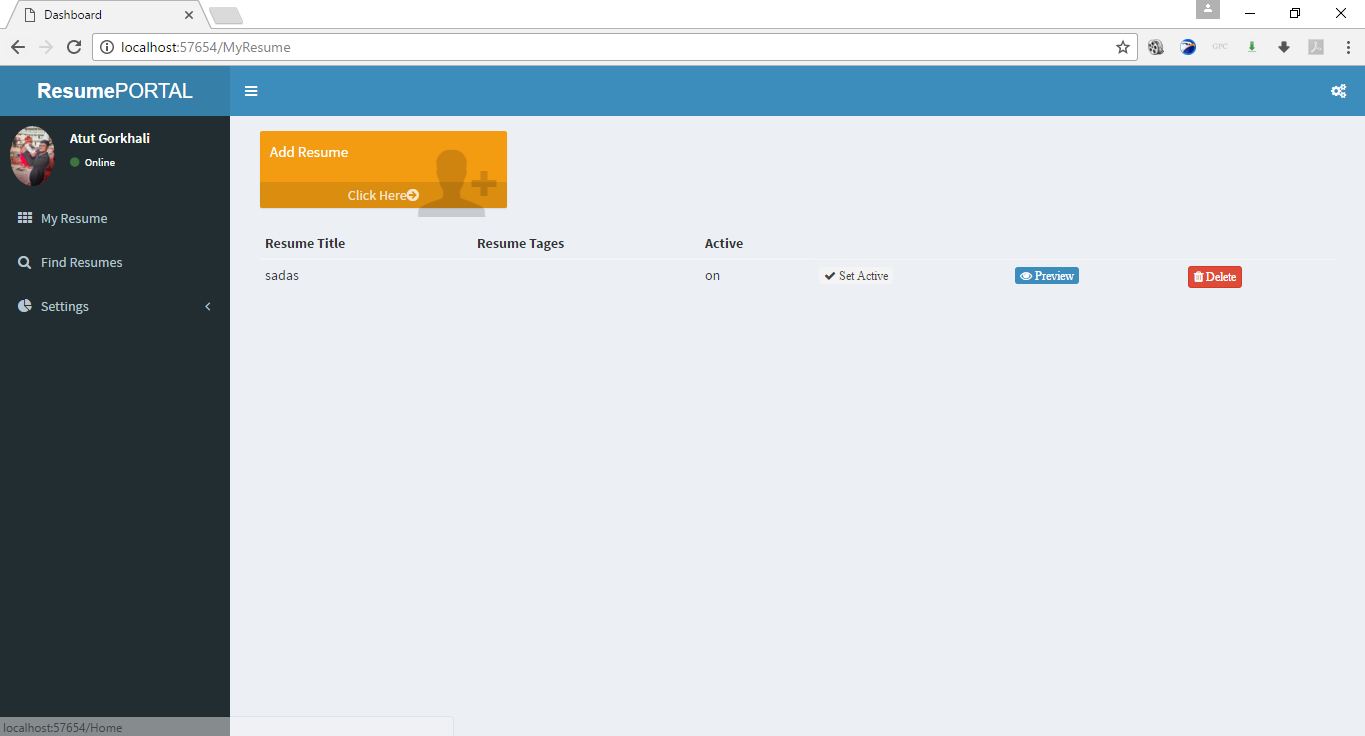


Figure 24 resume portal in chrome browser

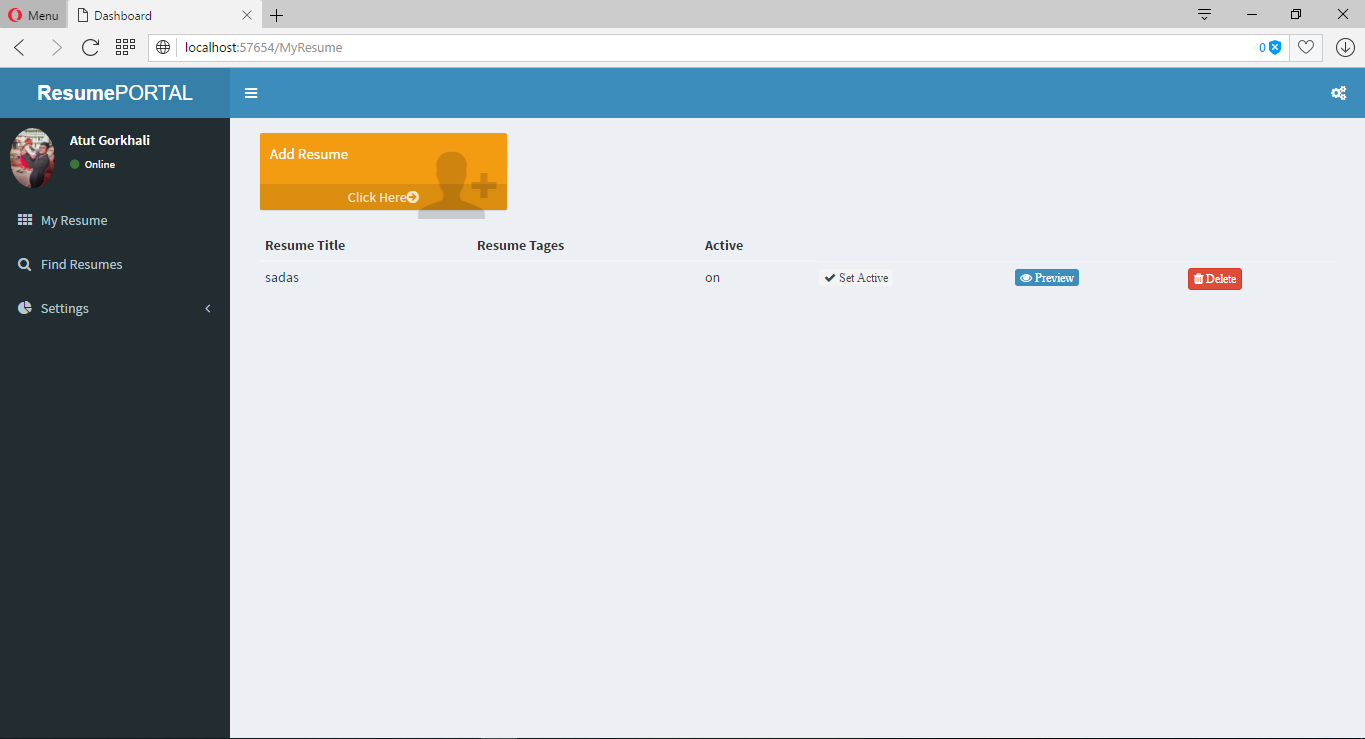


Figure 25 resume portal in Opera browser

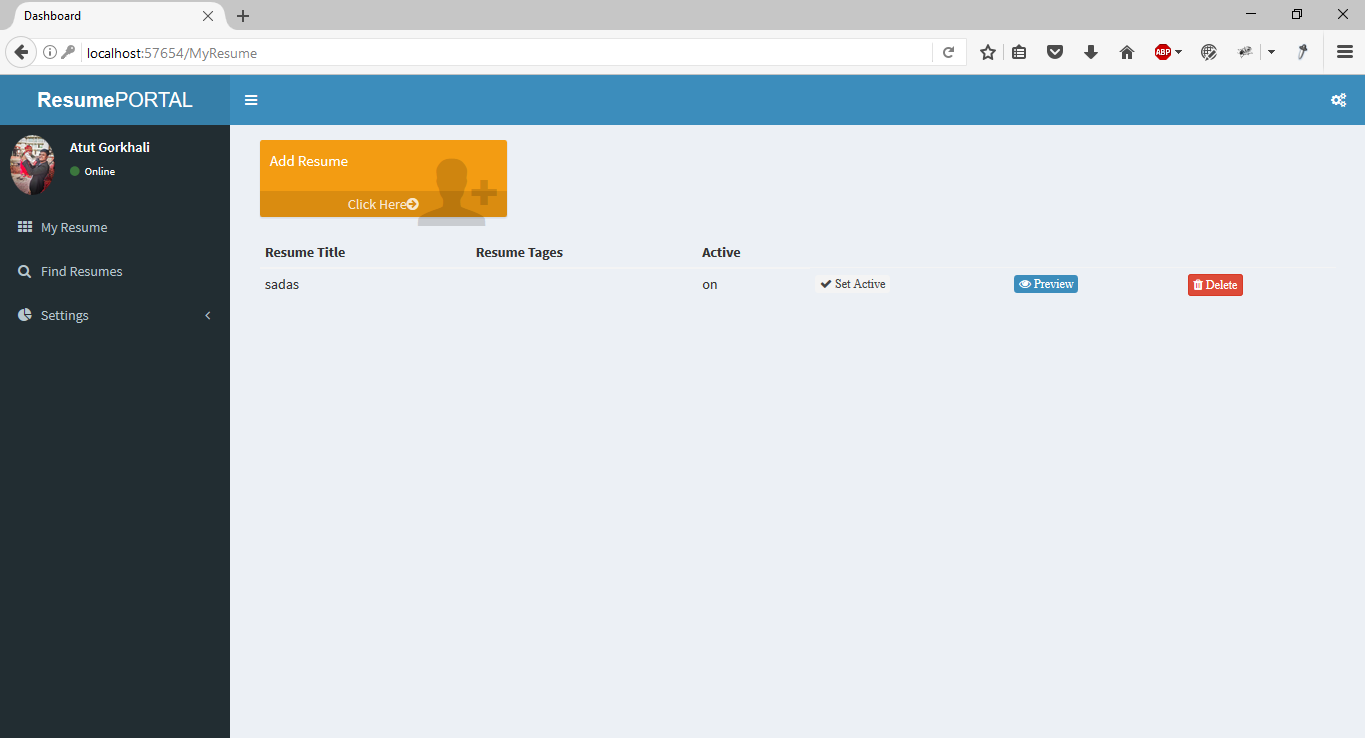


Figure 26 resume portal in Mozilla browser

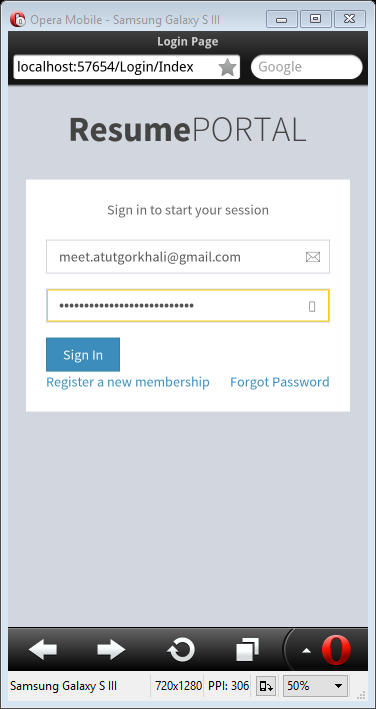


Figure 27 resume portal in mobile browser

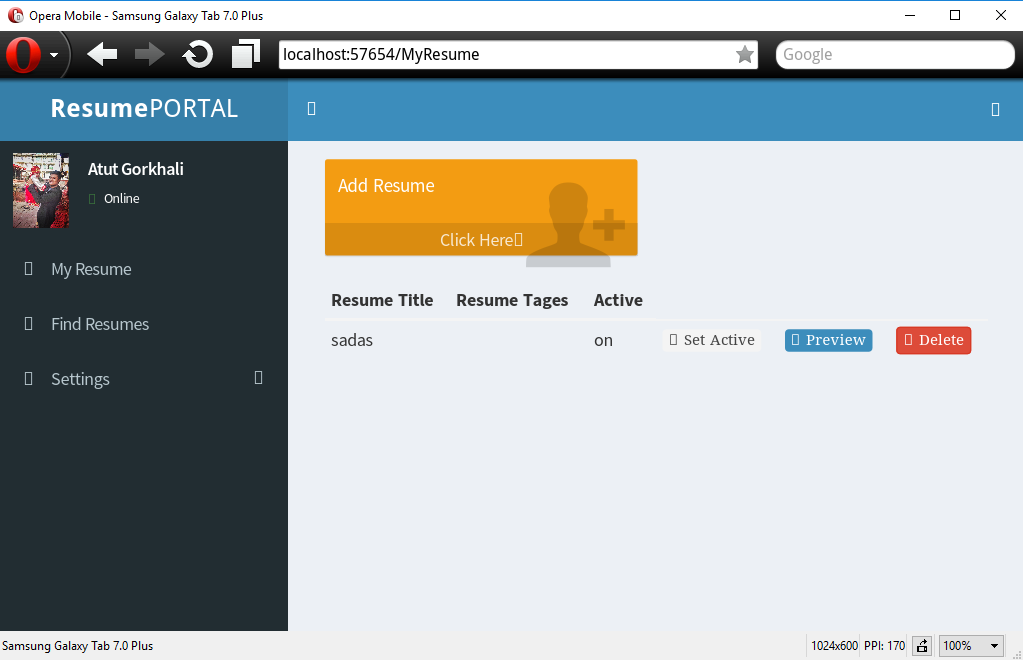


Figure 28 resume portal in Tab browser

### Validation Test Images

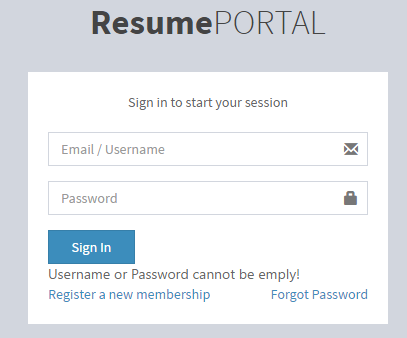


Figure 29 login form validation test

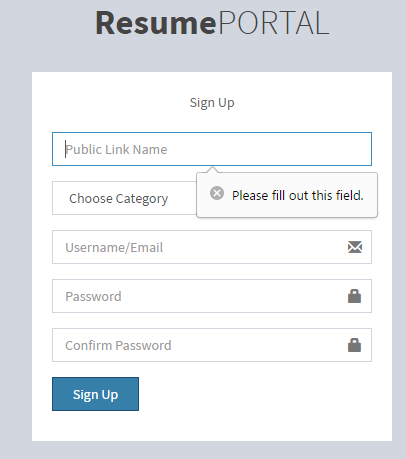


Figure 30registration form validation test

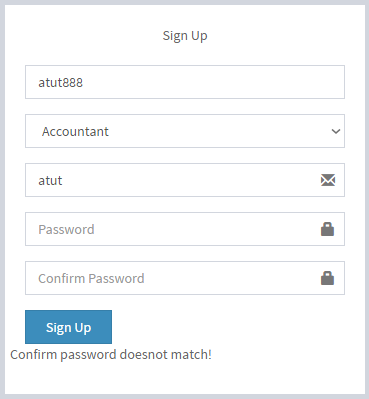


Figure 31registration form password validation test

## Appendix B

### Feasibility Study

#### Introduction

This feasibility section of report is prepared to understand the feasibility of project to be carried out based on various feasibility criteria. Before preparing system requirement specification (SRS) it is essential to analyze the feasibility to proposed system. Hence this document studies and evaluates the viability system to be developed based on available resources and technology. Once feasibility of proposed system is analyzed against alternative solution, system requirement specification document is prepared.

#### Purpose

The core purpose of this feasibility studies are as follows:

* Understand the technical feasibility of project
* Understand the schedule feasibility of project
* Understand the legal feasibility of project
* Understand the economic feasibility of project

#### Scope

This feasibility study would study feasibility of proposed project against various feasibility criteria. To carry out such study available resources such as technology or time will be analyzed against required resources for the project. Then submitted documents would be studied to understand legal feasibility of project.

#### Assumption and Constraints

There are various constraints that are going to apply on proposed project. Based on the available academic schedule, project must be delivered within three months. This includes analyzing requirements, planning, designing, developing, testing and documentation of project. Another key constraint of project is it should either use oracle database system or user C# technology during development. These constraints would be used during the feasibility evaluation of project.

#### Evaluation Criteria

Feasibility of a project is evaluated via examine the project against various evaluation criteria. For example, a project is technically feasible if all Technologies required for system are available in market. Feasibility criteria for this feasibility study are as follows.

* Cost Interface
* Schedule Interface
* Technology Interface
* Legal Interface

#### Proposed System

Development of online Resume Portal system using NoSQL technology is proposed as initial project proposal.

##### Description

In simplest words, this project would be an Online Resume builder that would utilize MongoDB technology as database solution. This project is essential as it would help on critical study of NoSQL technology over traditional relational database. If there is still time available after project completion, relational database solution would be developed and integrated with project. This would help to critically evaluate use of NoSQL as well as relation database in proposed project.

#### System Feasibility

Based on the submitted initial project plan for the project here are some high-level prediction of system requirements.

##### Technical Feasibility

|  |  |
| --- | --- |
| **High Level technical requirements (Estimation)** | |
| Computer with suitable configuration for coding, designing and documenting | Available [Own] |
| Visual Studio | Available [Free] |
| MongoDB database | Available [Free] |
| Web Publishing | Available [In Budget] |
| Internet | Available [Own] |

Table above describes the basic technical requirements for the proposed resume portal project at very highest level. This list includes very essential technical component required to complete the system without compromising the aims and objectives of project. All high-level components in this estimated list are accessible in market hence project would not suffer any technical difficulties during planning, development or implementation phase. Even though all components are available in the market, it is also important to consider the affordability to technology. However, most of the technologies required for the resume portal are either already owned or are easily available for free.

##### Schedule Feasibility

|  |  |
| --- | --- |
| **High Level Schedule requirements (Estimation)** | |
| Project Analysis | 15 Days |
| Design | 15 Days |
| Development | 30 Days |
| Testing / Debugging | 15 Days |
| Documenting | 10 Days |
| Deployment | 5 Days |
| Total | 95 +/- 10 Days |

Table above illustrates the very high level estimation of potential time requirement for the completion of project. This estimation is done without deep analysis hence may not be exactly accurate but would help to understand the schedule feasibility of the system. Estimation shows project would require roughly 100 Days even providing additional 10 days. This would allow more flexibility while planning actual schedule management. Meanwhile, time required for project is within the available schedule.

##### Economic Feasibility

|  |  |
| --- | --- |
| **High Level Schedule requirements (Estimation)** | |
| Computer System | Own (Free) |
| Internet | 6000 (1000 \* 6 Months) |
| Human Resource (Planner, developer and tester) | 15000 |
| Deployment | 4000 |
| Visual Studio | Free |
| MongoDB database | Free |
| RoboMongo | Free |
| Browsers | Free |
| Bit Bucket (GIT) | Free |
| Total | 20,000 |

Table above showcases the high-level cost estimation for the proposed resume portal system. Most of the required component are not required to be brought and can be available for free. However, project would require budget for human resources, internet and deployment of the system. During the fact findings for the project and constraint/assumption analysis project is understood to be allocated with RS 30,000. The high-level cost estimation is well below the allocation budget.

#### Feasibility of the system

During the feasibility analysis of proposed resume portal, economic requirement, schedule requirement and technical requirement for development of product and successful completion of project was analyzed. Based on the very high level findings it is understood that resume portal is feasible project against each feasibility testing criteria.

## Appendix C

### Implementations Images

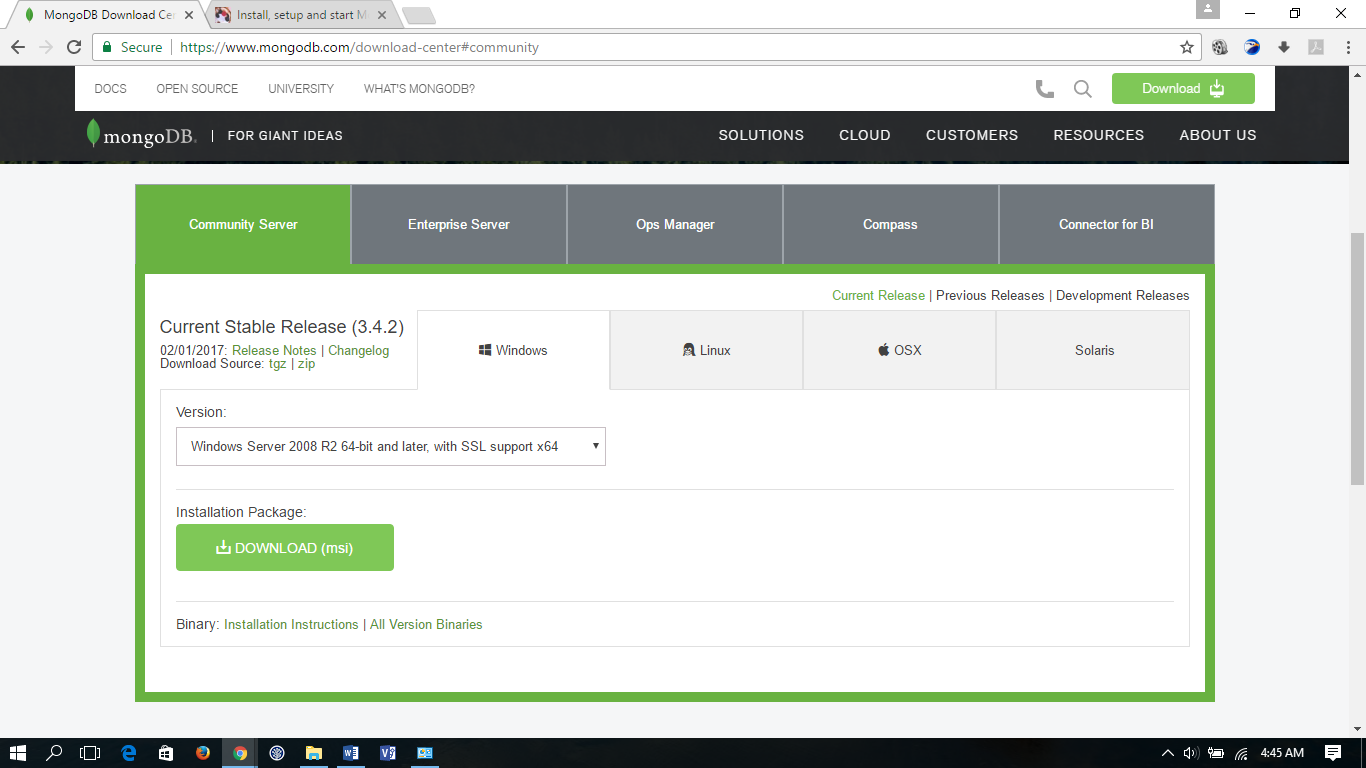


Figure 32Downloading MongoDB

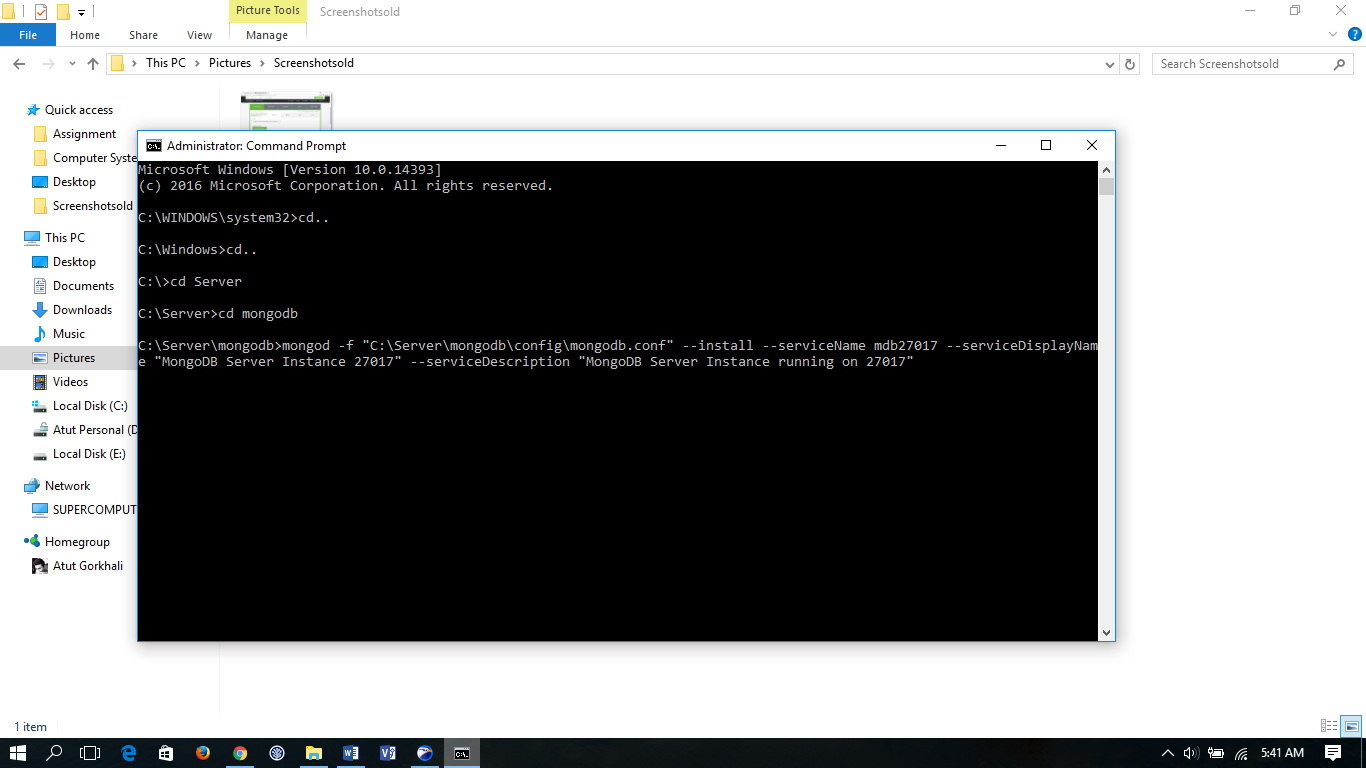


Figure 33 Creating MongoDB instance as windows Service

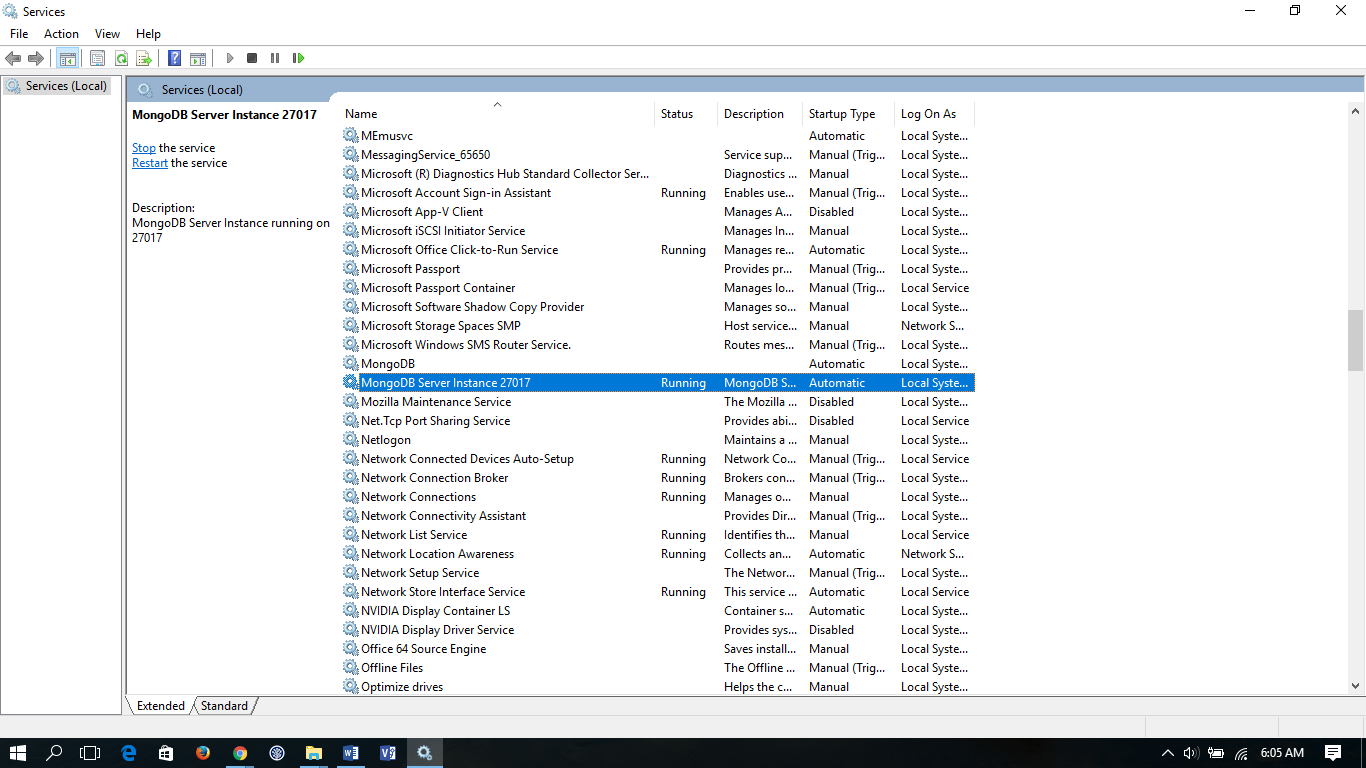


Figure 34MongoDB instance as windows service

## Appendix D

### Meeting Record Sheets

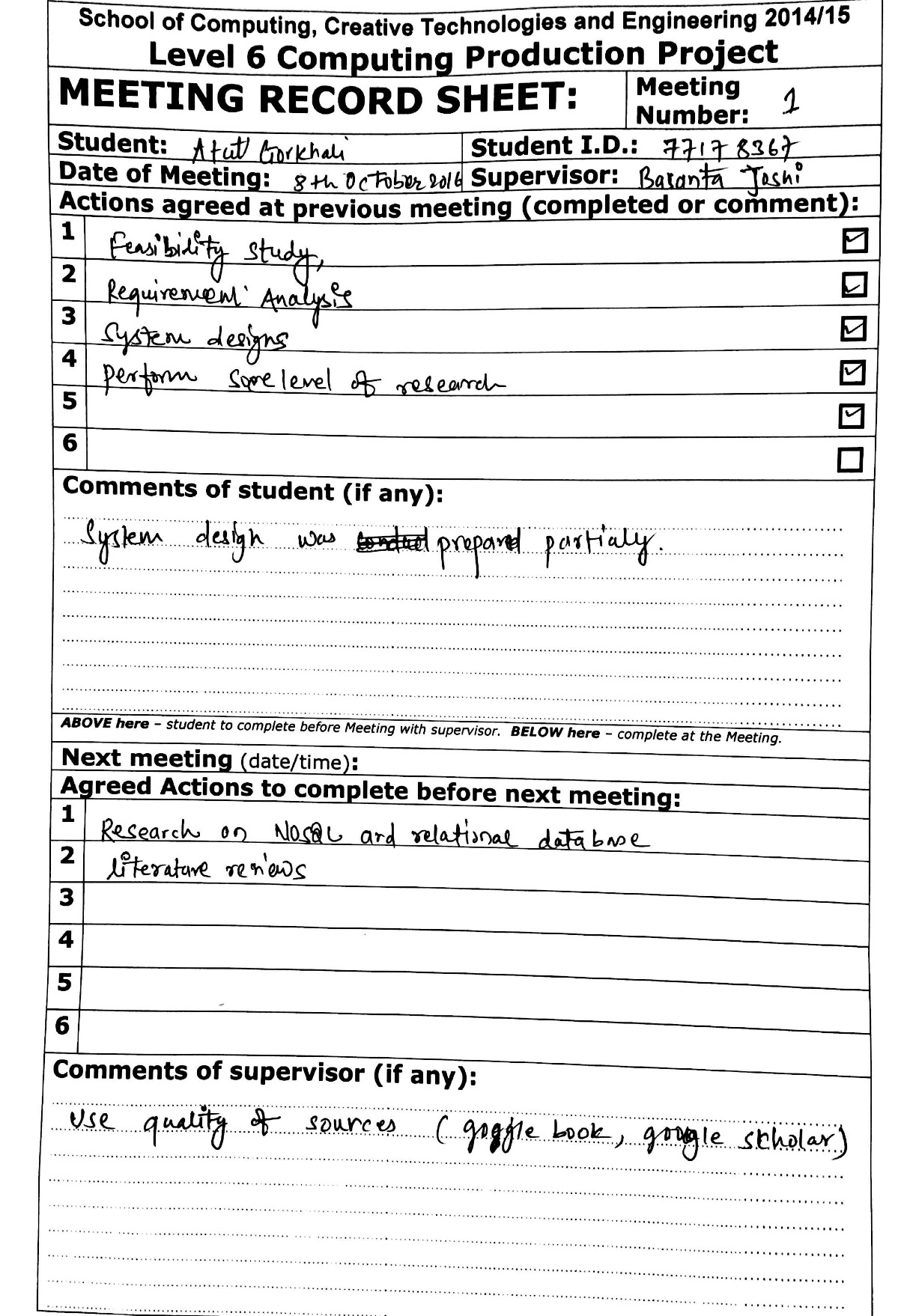


Figure 35Meeting record sheet 1

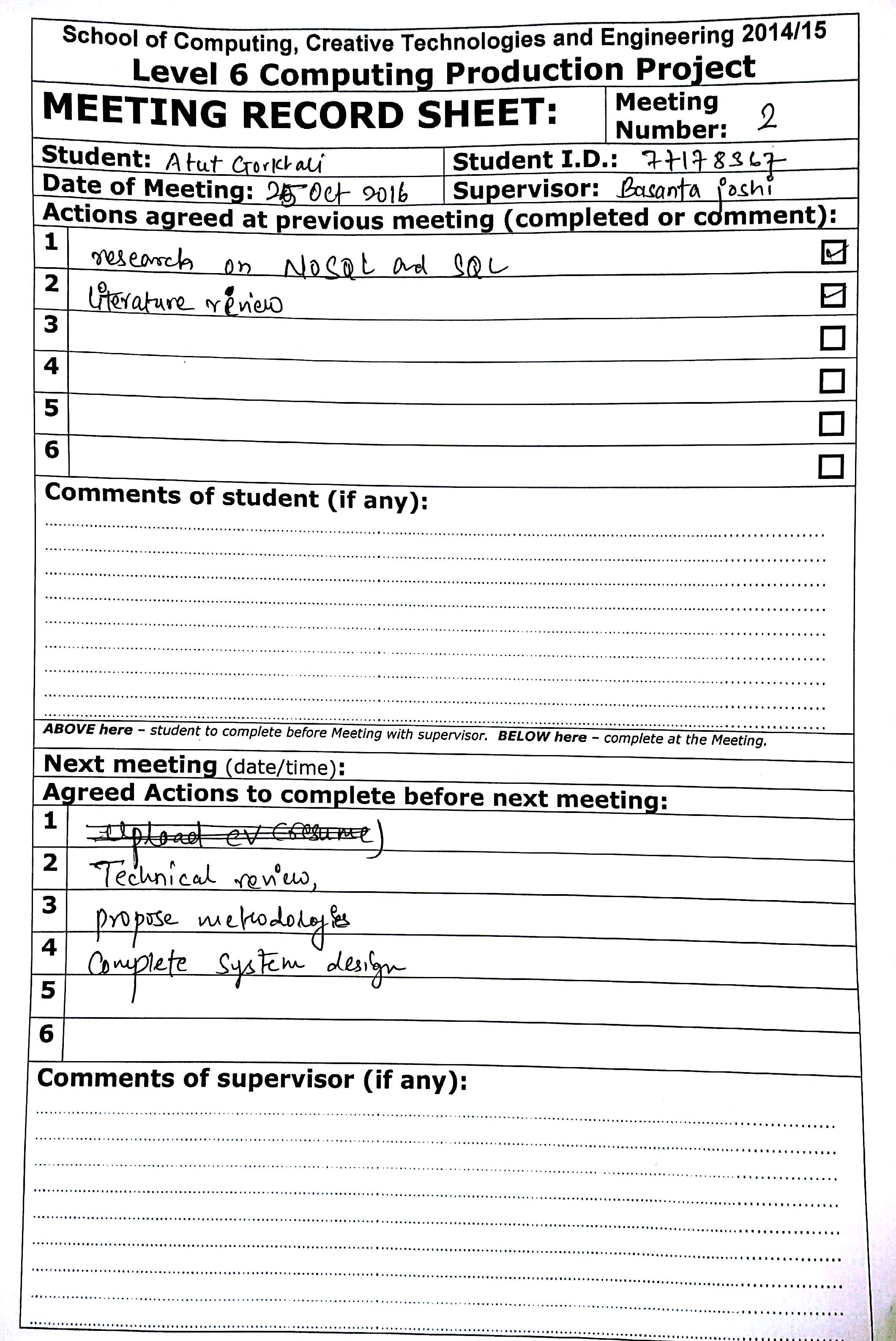


Figure 36Meeting Record sheet 2

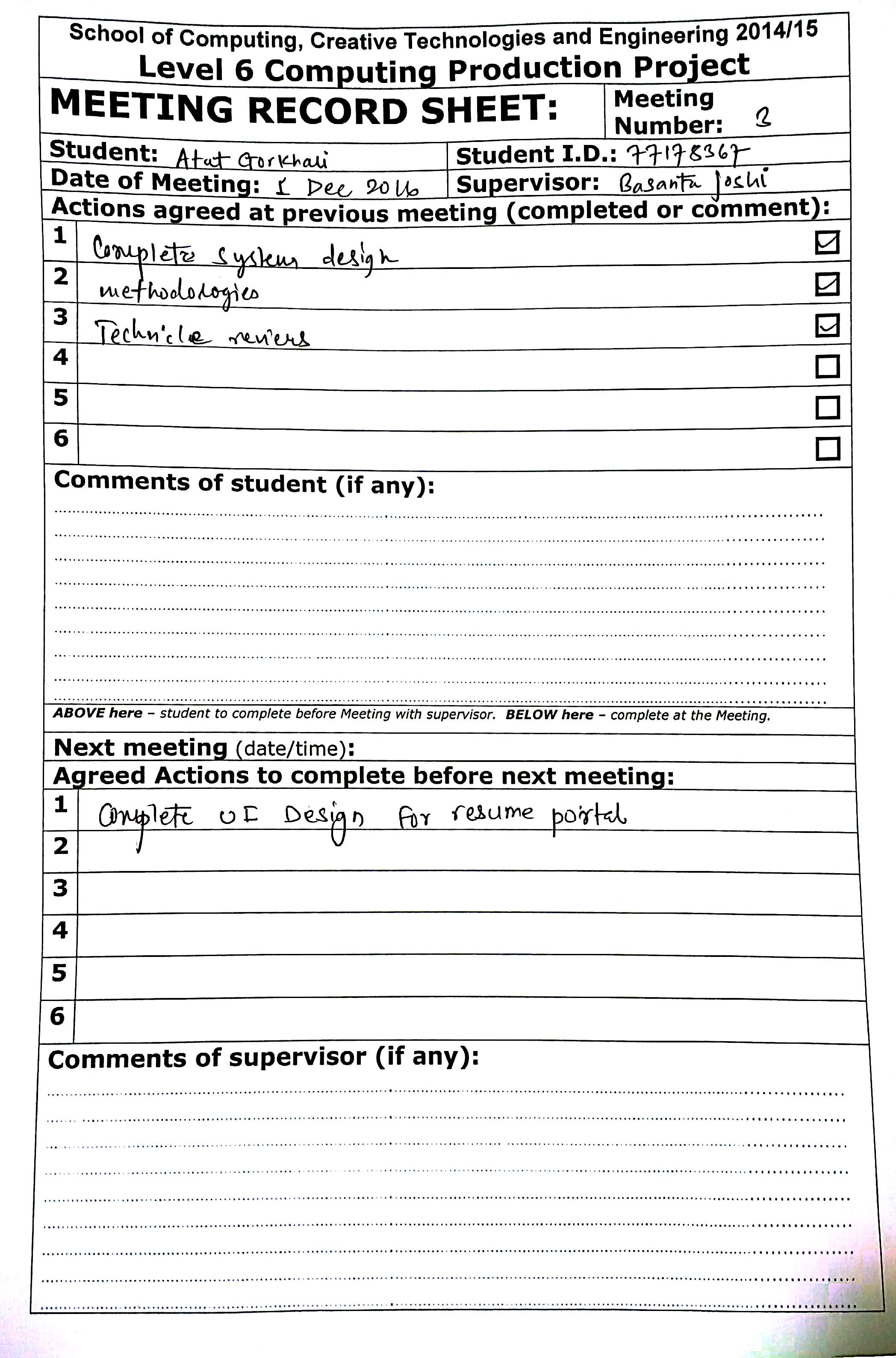


Figure 37Meeting Record sheet 3

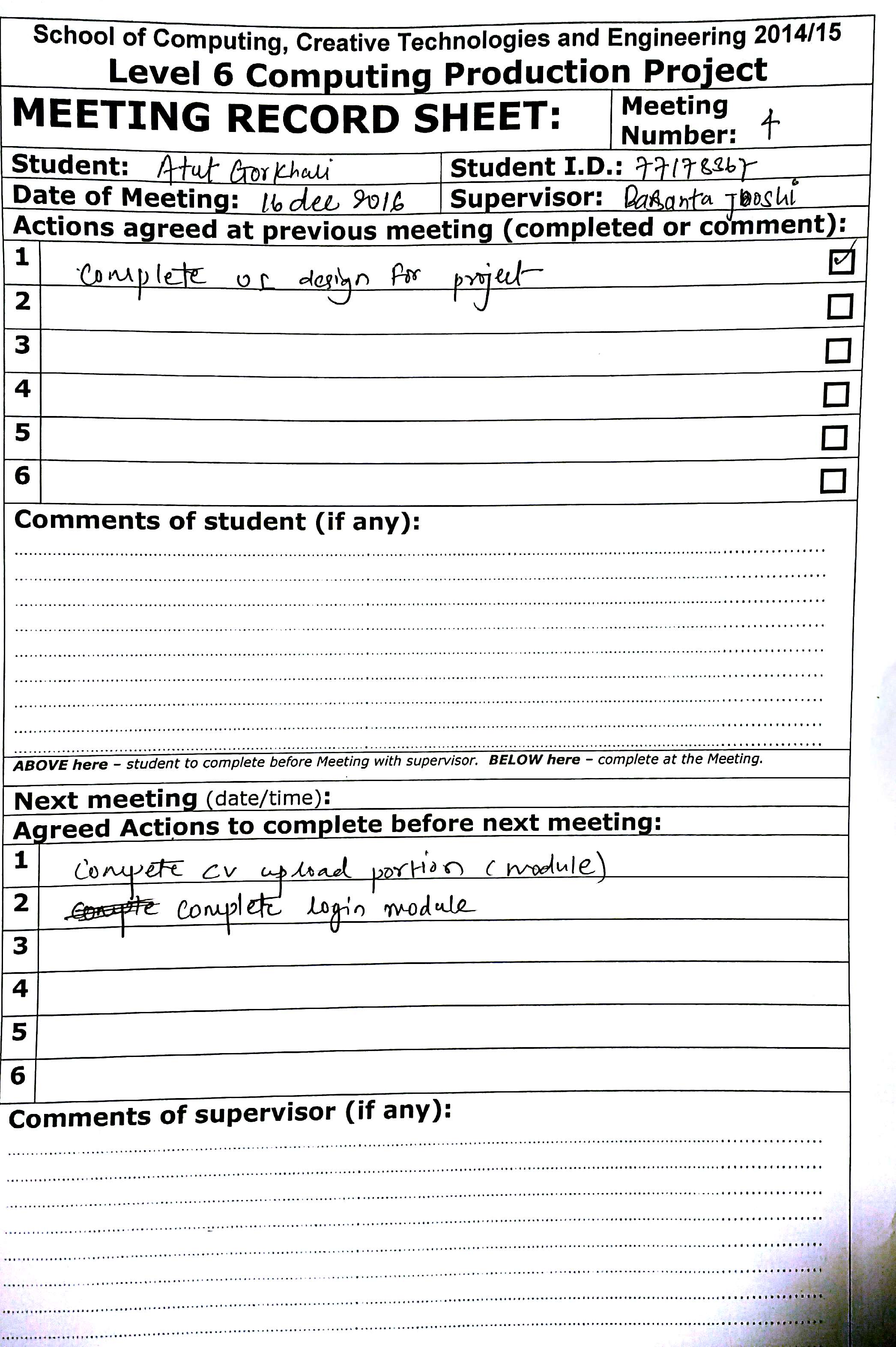


Figure 38Meeting Record sheet 5

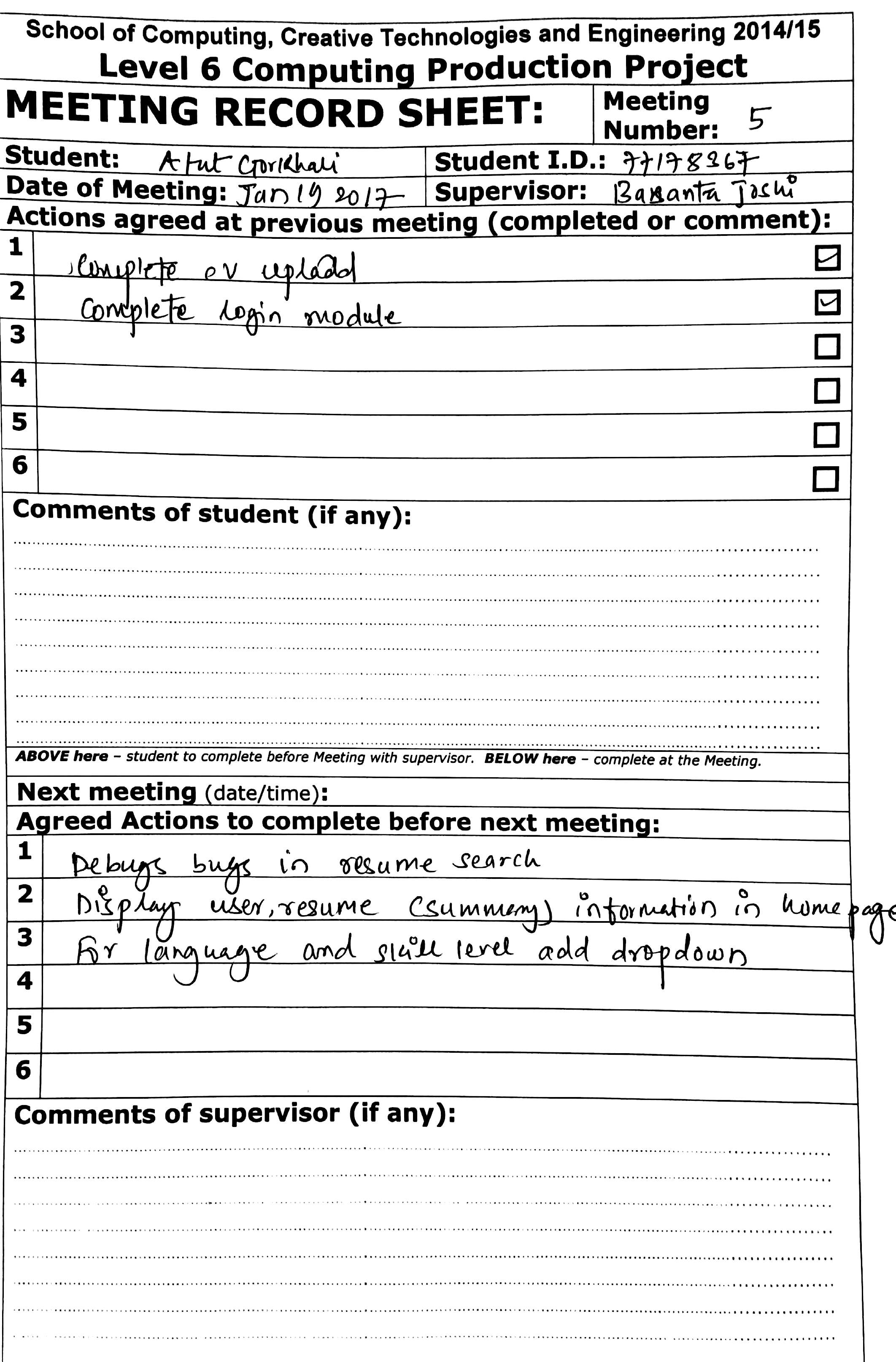


Figure 39Meeting Record sheet 5

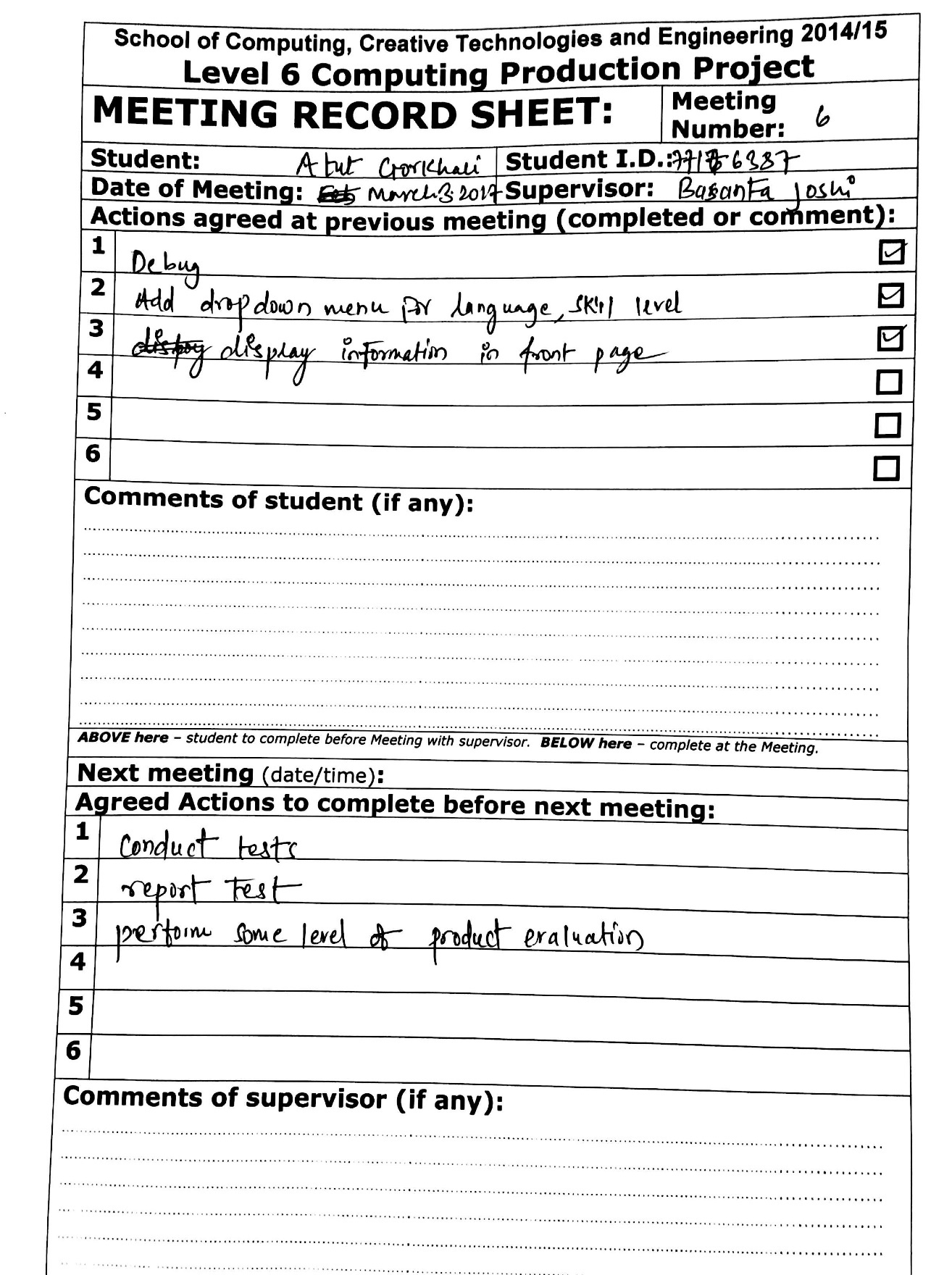


Figure 40Meeting Record sheet 6

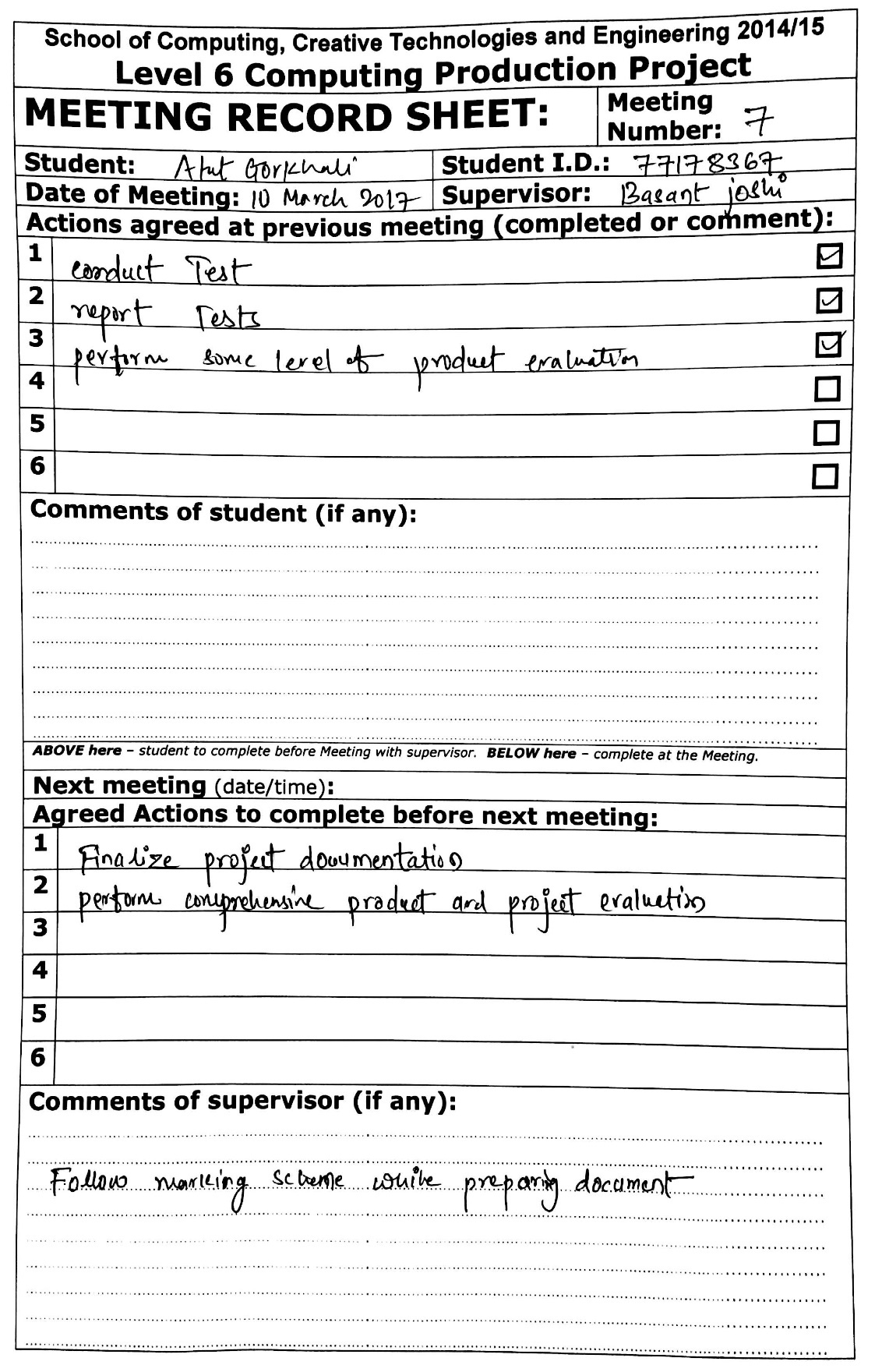


Figure 41Meeting Record sheet 7

## Appendix D

### Git Repository Evidence

Resume Portal system was version controlled in bit bucket system. This acted as backup solution for project. In any case of project loss, progress could be retrieved from version control.

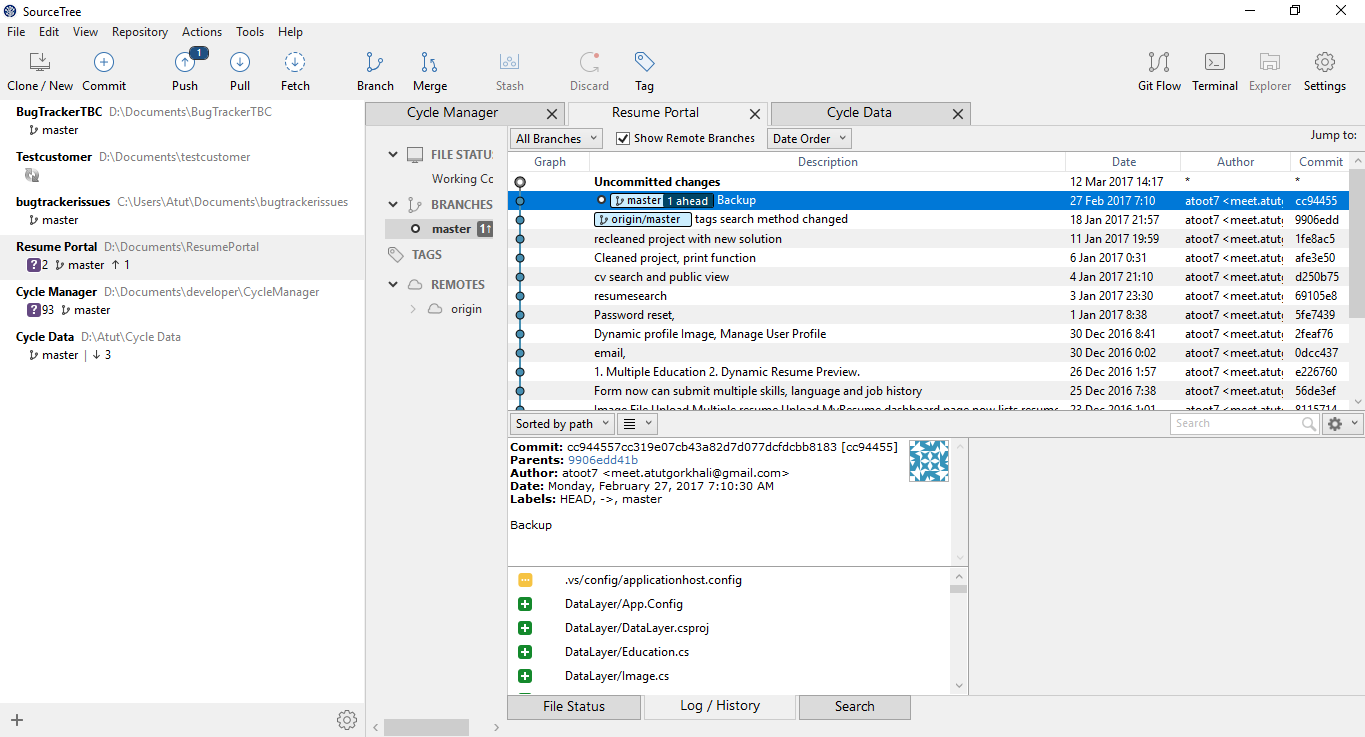


Figure 42 Version Control client (Source Tree)

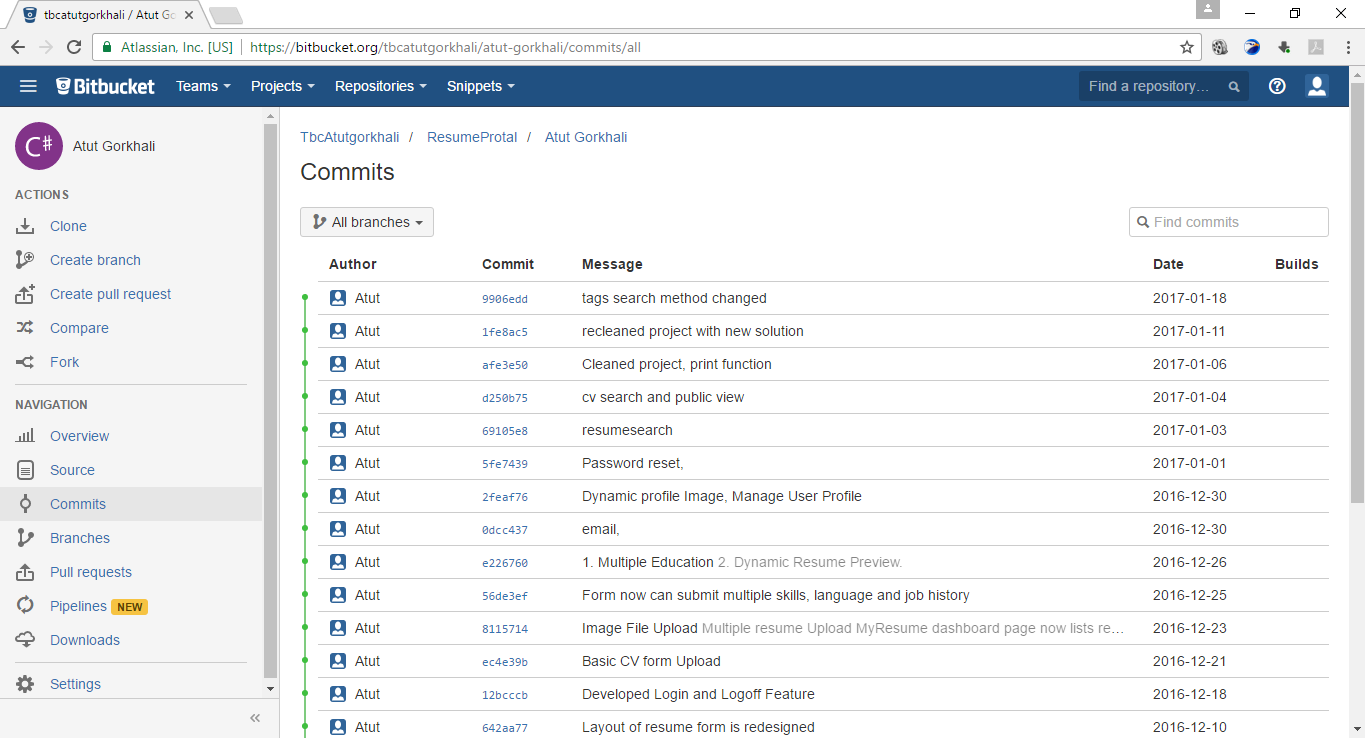
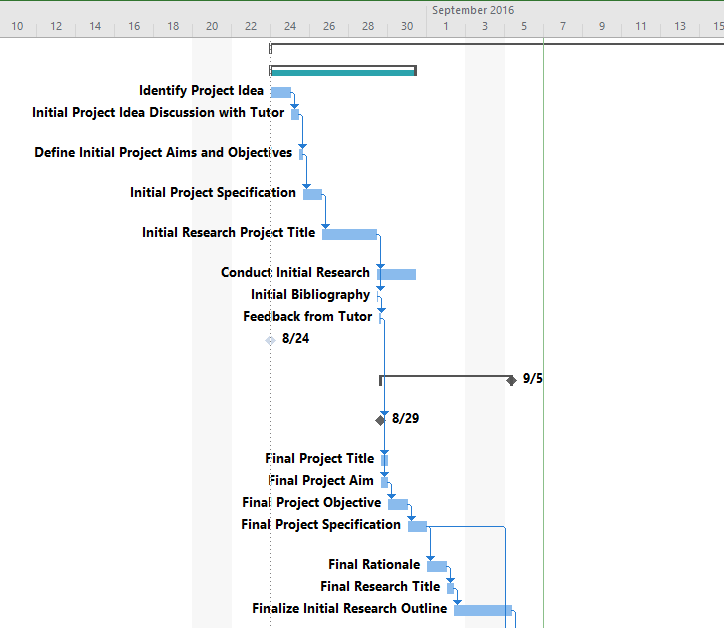
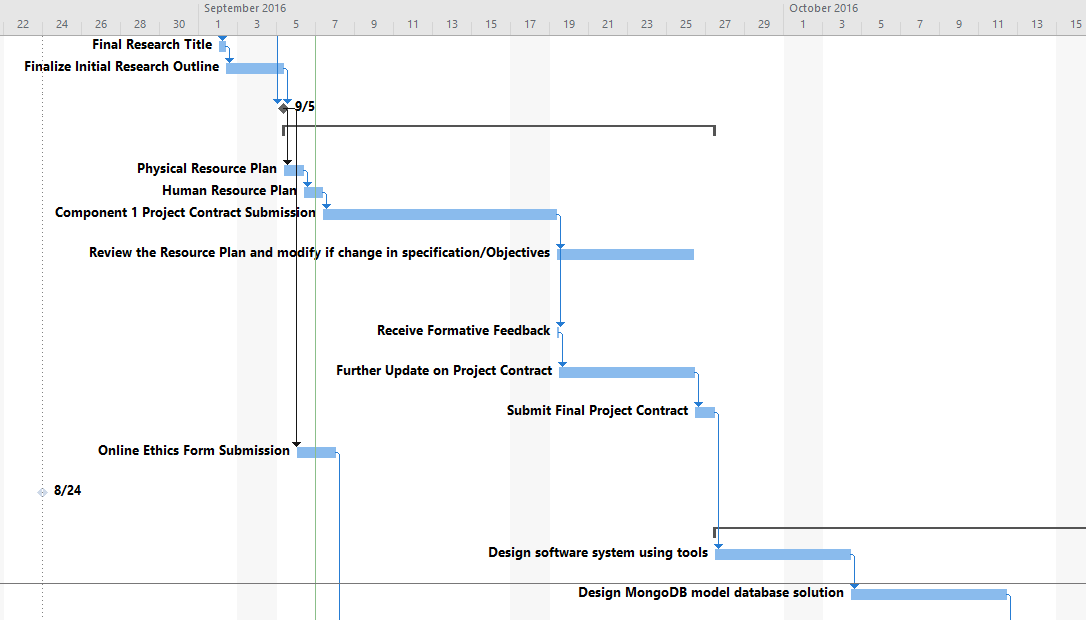


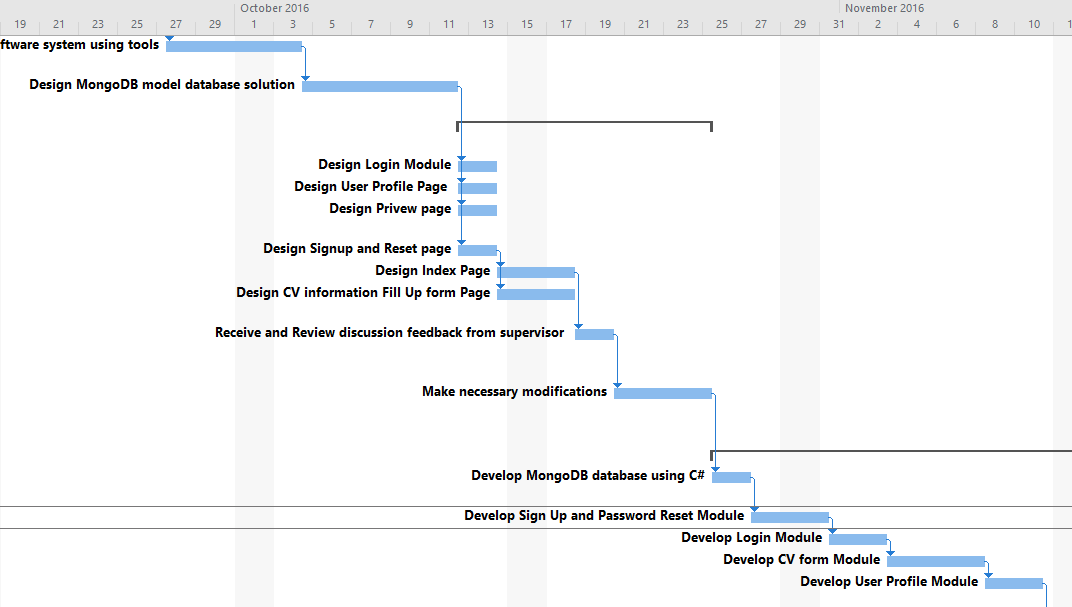
Figure 43 Version Control Server (Bit Bucket)

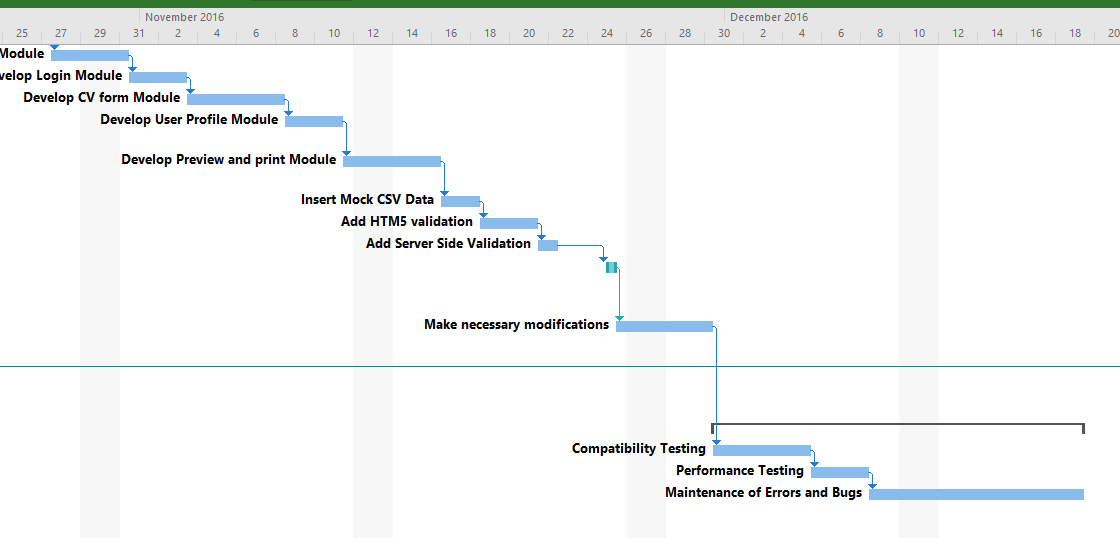
## Appendix E

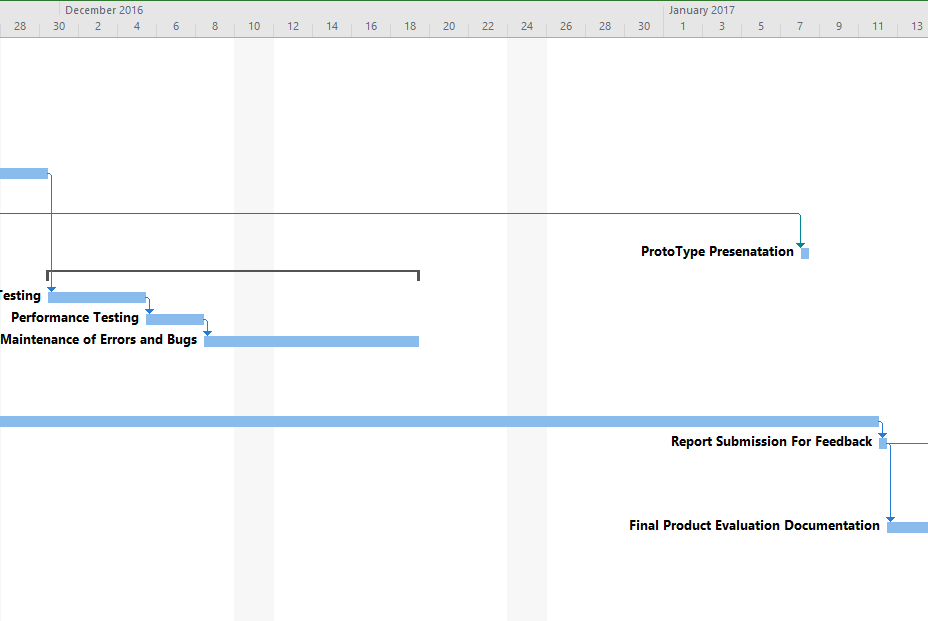
##### Gantt Chart

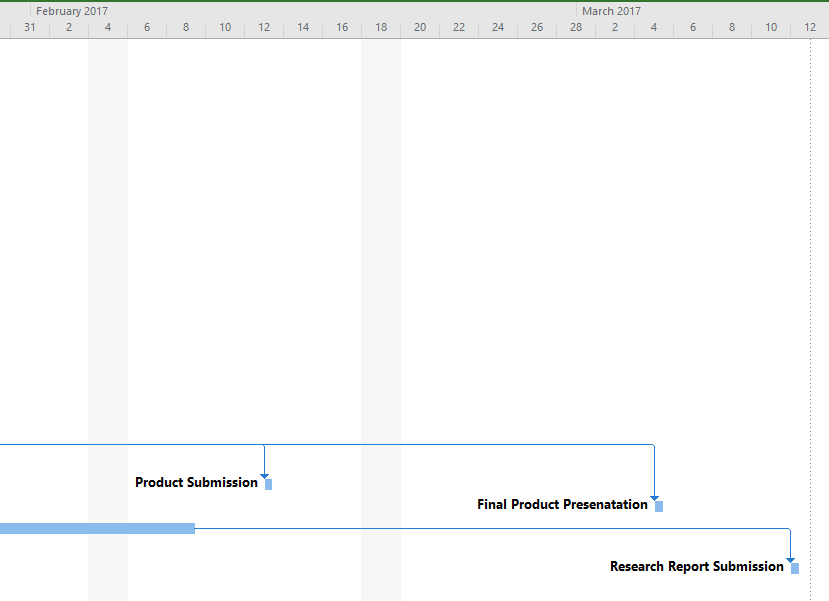












## Appendix F

#### C# with MongoDB driver

public async Task<bool> DoUpsertAsync(ResumeModel resume)

        {

            IMongoCollection<ResumeModel> collection = ConnectionBl.MongoDbConnection();

            var filter = Builders<ResumeModel>.Filter.Eq("\_id", ObjectId.Parse(resume.\_id.ToString()));

            var result = await collection.ReplaceOneAsync(

                filter: new BsonDocument("\_id", resume.\_id),

                options: new UpdateOptions

                {

                    IsUpsert = true

                },

                            replacement: resume);

            if (result.ModifiedCount > 0)

            {

                return true;

            }

            else

            {

                return false;

            }

        }

#### C# with Relational Database

public bool SaveSQLCV(ResumeDetails model)

        {

            try

            {

                using (var ent = new ResumePortalDbEntities())

                {

                    //assuming logged in user for testing purpose

                    var assumedUserId = 1;

                    var name = model.FullName.Split(null);

                    //saving cv

                    var cv = new DataLayer.ResumeDetail()

                    {

                        FirstName = name.FirstOrDefault(),

                        LastName = name.LastOrDefault(),

                        DateOfBirth = model.DateOfBirth,

                        Phone = model.Phone,

                        Website = model.Website,

                        UserId = assumedUserId

                    };

                    ent.ResumeDetails.Add(cv);

                    ent.SaveChanges();

                    //get resume id

                    var resumeId = cv.ResumeId;

                    //saving skills

                    var skillList = model.Skills;

                    if (skillList != null && skillList.Any())

                    {

                        foreach (var skill in skillList)

                        {

                            var skilldata = new DataLayer.Skill()

                            {

                                ResumeId = resumeId,

                                LevelId = int.Parse(skill.Level),

                                Name = skill.Name,

                            };

                            ent.Skills.Add(skilldata);

                            ent.SaveChanges();

                        }

                    }

                    //saving languages

                    var languageList = model.Languages;

                    if (languageList != null && languageList.Any())

                    {

                        foreach (var language in languageList)

                        {

                            var languageData = new DataLayer.Language()

                            {

                                ResumeId = resumeId,

                                LevelId = int.Parse(language.Level),

                                Name = language.Name,

                            };

                            ent.Languages.Add(languageData);

                            ent.SaveChanges();

                        }

                    }

                    //saving job history

                    var jobHistoryList = model.JobHistories;

                    if (jobHistoryList != null && jobHistoryList.Any())

                    {

                        foreach (var job in jobHistoryList)

                        {

                            var jobData = new DataLayer.JobHistory()

                            {

                                ResumeId = resumeId,

                                Company = job.Company,

                                Description = job.Description,

                                IsActive = job.IsActive,

                                Location = job.Location,

                                StartDate = job.StartDate,

                                EndDate = job.EndDate,

                                Title = job.Title

                            };

                            ent.JobHistories.Add(jobData);

                            ent.SaveChanges();

                        }

                    }

                    //saving job history

                    var educationList = model.EducationInfos;

                    if (educationList != null && educationList.Any())

                    {

                        foreach (var edu in educationList)

                        {

                            var eduData = new DataLayer.Education()

                            {

                                ResumeId = resumeId,

                                Institute = edu.Institute,

                                Description = edu.Description,

                                IsActive = edu.IsActive,

                                Location = edu.Location,

                                StartDate = edu.StartDate,

                                EndDate = edu.EndDate,

                                Title = edu.Title

                            };

                            ent.Educations.Add(eduData);

                            ent.SaveChanges();

                        }

                    }

                    // saving Image info

                    var imageInfo = new DataLayer.Image()

                    {

                        Name = model.Image.Name,

                        ResumeId = resumeId

                    };

                    ent.Images.Add(imageInfo);

                    ent.SaveChanges();

                }

                return true;

            }

            catch (Exception)

            {

                return false;

            }

        }