1. Since the project is a specific application in Combinatorial Block designs, I will begin with briefly explaining what they are
2. Following that I will introduce the project
3. Then I will continue with describing the Ext Rep of block designs and what I had to do with it
4. Then the database of combinatorial designs that I built
5. Followed by an overview of the web interface to the system
6. I will also discuss the deployments scheme
7. and finally I will conclude with an overview of the system and the general conclusions of the project

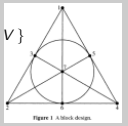
Since my Project is an Application in Combinatorial Block Designs, we should start by explaining What are Block Designs?

1. A block design is essentially composed of two sets, a set of points V and a set of blocks B. The design itself is a Multiset of subsets of the point set

What are Block Designs?

- Multiset of subsets of points

$$D = (B, V)\,;\; B = \{\, b \mid b \subset V \,\}$$



- Example (Fano Plane)

```
V = [0, 1, 2, 3, 4, 5, 6]
B = [ [0, 1, 2], [0, 3, 4], [0, 5, 6], [1, 3, 5],
      [1, 4, 6], [2, 3, 6], [2, 4, 5] ]
```

1. looking at an example, we have a design with given point set V, composed of seven points. The design is subsets of the points in V, these subsets are called blocks. This specific design has 7 blocks each of which is composed of 3 points. This design is actually a famous design labelled as the Fano Plane

What are Block Designs?

- Multiset of subsets of points

$$D = (B, V)\,;\, B = \{\, b \,|\, b \subset V \,\}$$

- Example (Fano Plane)

```
V = [0, 1, 2, 3, 4, 5, 6]
B = [ [0, 1, 2], [0, 3, 4], [0, 5, 6], [1, 3, 5],
      [1, 4, 6], [2, 3, 6], [2, 4, 5] ]
```

- Useful in experimental design, finite geometry, software testing, cryptography, error correcting codes.

1.
   - block designs are utilized in various fields including the design of scientific experiments.
   - Using the example that introduced me to block designs, the Fano Plane is an optimal design for scientifically testing 7 brands of ice cream.
   - It is not ideal to attempt to test all 7 at the same time...
   - Divide the experiment into smaller experimental units, where within each one we can pairwise compare the elements.
   - Moreover, The properties of this design inform us that after performing the smaller experiments we can reach the goal of comparing all the products.
2. Block designs are also useful in finite geometry, for example the image in the top right corner is a geometric representation of the Fano Plane

What are Block Designs?

- Multiset of subsets of points

$$D = (B, V)\,;\; B = \{\, b \mid b \subset V \,\}$$

- Example (Fano Plane)

```
V = [0, 1, 2, 3, 4, 5, 6]
B = [ [0, 1, 2], [0, 3, 4], [0, 5, 6], [1, 3, 5],
      [1, 4, 6], [2, 3, 6], [2, 4, 5] ]
```

- Useful in experimental design, finite geometry, software testing, cryptography, error correcting codes.
- Hamming codes were discovered as a block design 5 years earlier [Cam08].

1. Designs are useful in general, in areas that are related to combinatorics. Other areas include software testing, to test combinations of input, cryptography and error correcting codes.
2. A brief history byte, R.A. Fisher discovered the Hamming codes as factorial designs five years before R. W. Hamming found them in the context of error correction.

My project deals with these combinatorial block designs in three major phases

1. First I will discuss the Ext Rep, and more particularly the implementation of $v3$ of the specification.
2. Followed by an overview of the Comb. Bl. Des DB
3. I will then describe the Implementation of the Web Interface to the system.

There are a few factors that motivated the initiation of this project

1. Historically were located by searching through the publications in which the particular design was published. Also some publications only displayed the blocks of the design, or only the properties that were needed within their research.
2. More generally a public central repo for block designs was not available until an initiative through the Queen Mary University, and more specifically DesignTheory.org began creating block designs and storing them as simple files for the public consumption.
3. The designtheory.org collection which contains around 2.5M designs was published in Ext Rep $v2$ (XML based representation), which was later seen to not be suitable for mathematical data and thus triggering the Implementation of Ext Rep $v3$, which we will be discussing.

- Method of communicating designs
- Communicating design properties
- Invented at the Queen Mary University of London
- Platform Independent
- User Agents = Human | Software
- Ext Rep *v*2 published mid 2004

So, what is the External Representation of block designs

1. it is a means of communicating block designs. Block designs as mentioned earlier are simply a set of blocks. Therefore the set of blocks is the minimum required to communicate a design
2. However, there are also properties associated with each block design; for example the statistical robustness properties are valuable in experimental design significance when block lost
3. The concept was invented by a group of researchers at QMU, which include Dr. Dobcsányi
4. designed to be platform ind. such that it facilitates communications between all types of user agents
5. User agents were categorized as both software and human beings, thus the Ext Rep has to be readable.
6. *v*1.1 published in 2003, and was updated to *v*2 in 2004

```
<blocks ordered="true">
    <block><z>0</z><z>1</z><z>2</z></block>
    <block><z>0</z><z>3</z><z>4</z></block>
    <block><z>0</z><z>5</z><z>6</z></block>
    <block><z>1</z><z>3</z><z>5</z></block>
    <block><z>1</z><z>4</z><z>6</z></block>
    <block><z>2</z><z>3</z><z>6</z></block>
    <block><z>2</z><z>4</z><z>5</z></block>
</blocks>
```

(a) v2 [CDMS03]

```
"blocks" : [
  [0, 1, 2], [0, 3, 4], [0, 5, 6],
  [1, 3, 5], [1, 4, 6], [2, 3, 6],
  [2, 4, 5]
]
```

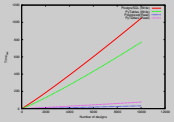
(b) v3 [DN08]

Figure: *blocks* in ExtRep versions

There were some shortcomings of the Ext Rep *v*2

1. – SGML suits textual information, as we see its success on the web in the form of X-HTML.
   – XML lacks the support for basic datatypes, and therefore puts a burden on the implementation to define datatypes
   – These extra definitions caused extra layers of XML tags making the design quite unreadable
2. JSON better suited block designs with the availability of the basic datatypes int, float, and string. JSON also has lists and objects (key, value pairs) which allowed a much simpler form for the design. See figure
3. gain advantages of JSON over XML some structural changes were made to the Ext Rep, one small example “block”
4. As a functionality enhancement the List invariants which was left out in previous extrep version for future consideration, was defined.

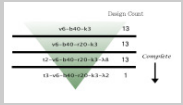
1. Since this change breaks compatibility, Python scripts were written to convert designs from v2 to v3. Also the collection being converted contained over 2.5M designs (hard to do manually). The reason the script was in Python because primary language for this project.
2. Python JSON parsers were available; However,
   - none allowed streaming, and some of the files are too big to load in memory.
   - Also the available parsers forced a particular internal representation which did not maintain the ordering of entries in the document.
   - I implemented A Python wrapper around the YAJL C library This wrapper allowed for a callbacks based parser that is flexible enough for my needs.



A database system was needed to store designs such that they can be searchable via any of the parameters. There were many options for a dbms after looking at the various option the battle was between these two.

1. HDF5 is leading popular HDBMS for storing scientific data. Advantages of using HDF5 was it has a nice wrapper in Python.
2. PostgreSQL is similarly a leading RDBMS. The reason PSQL was chosen over alternatives like Sybase or Oracle, is because PSQL is free (open source). Over SQLite and MySQL because array datatype.
3. A set of prototypes and experiments were implemented to see which DBMS is more suitable for my application.
4. Experiment based on the initial prototypes showed PostgreSQL to be faster at querying returning results
5. Moreover, the internal representation for HDF5 prototypes was more complex for substructures like blocks.

1. The Ext Rep Structure which holds a design and its properties can be represented as a Tree. However, as mentioned, the sub-trees which hold the properties of the design are optional.
2. Each of the optional sub-trees was put in a table
3. FKs linked Sub-trees with parent nodes
4. Repeated nodes, represented as JSON lists were simply many to many relationship with the parent node
5. Software was written to map a JSON design to the DB structure and back to a JSON structure



1. The flexibility of the JSON parser allowed callbacks to generate Object Relational Mapping objects. JSON $\rightarrow$ Object $\rightarrow$ Database
2. The designthoery.org collection contained 3,000 designs that were isomorphic to one another. Software was written to filter out these designs from the database, s.t. the db contains only pairwise non-isomorphic designs
3. Designs are categorized based on their basic parameters, which are t, v, b, r, k, λ. The original file collection was organized, such that each file contained designs from the same category. This information was placed in the database. The design categories are hierarchical, and therefore a design is often in more than one category ...

- Encapsulated Design
- Query Language
- User/Session Management

```
{
  "designs" : [{
    "statistical_properties"  :  {
      "canonical_variances"  :  {
        "value"  :  {
          "canonical_variance": [
            ["<", 0.34], [">", 0.22]
          ]
        }
      }
    }
  }]
}
```

Figure: Query Example

1. Query Engine was implemented such that it was standalone. Thus the interaction to the database was hidden behind the query engine, and design retrieval happens via the Query Engine
2. To complete the encapsulation a method of specifying the search query was needed. I didn't think its a good idea to force the users to learn SQL, so I decided to build a simple query language.
3. Currently I am the only user of the Query Engine, but the way it was designed was to allow for the new Buzz "Server/vice oriented architecture" and would be used directly in the future.
4. To complete the black boxed query engine, a user management system was implemented into the Query Engine. A user wanting to issue queries must first ask for a session id, the following queries or result set browsing would are managed based on this session id

1. A python framework was to be chosen to easily allow integration with the other pieces of the system (i.e. Parser, Query Engine). A few were looked at and Pylons was chosen.
2. It was not a good idea to force users wanting to use the query engine to generate the query language syntax by hand. So three different methods to generate queries were designed
3. The first being through the summary table, which shows all the categories in the database and how many designs are available per category
4. Encapsulating the Query Engine allowed for an easy and simple interface with the Web Application.

1. Second method is using the small form located above the summary table which allows searching via the basic parameters of the design t, v, b, r, k, λ

1. The last method, is through the advanced search form, which allows querying via the entire ExtRep properties. You can click ... through the levels, reaching the leaf where the criteria is to be entered, and you can add the value to the query.

1. To display designs on the internet multiple techniques were attempted, but in the end the decision was to display the designs in their Ext Rep form as we can see.
2. It reminds me of a recent comment that "I should eat my own dogfood". The Ext Rep was designed such that it is human readable, and therefore it should be suitable to display directly.
3. Moreover, the Ext Rep structure in whole or parts of it are valid representation for a few languages and systems like matlab, mathematica, R, GAP and Python

- Designed for Unix
- Apache2 w/ Mod Wsgi
- Load testing

1. The system was made such that it is easily deployed on unix like systems, such as the variety of flavours of GNU/Linux
2. From the variety of options, Apache2 was chosen as the web server to run the Python web application through. This was done through the WSGI Apache module, which implements the Web Server Gateway Interface which is the current Python standard
3. Three experiments were run to test the breaking point of the system. Each of the experiments simulated different usage strategies of the system through the web interface. I found that the system running on a single 2.4 Ghz machine with 1G of ram, can handle between 700 to 1000 simultaneous users per hour

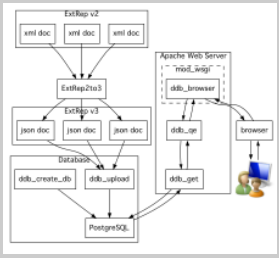
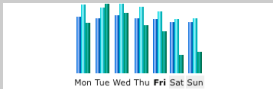


Figure: Design DB Overview

1. We saw the details of the system, to stand back and look at the system these are he various modules and how they interact to help the users find and retrieve combinatorial block designs



Day	Pages	Hits	Bandwidth
Mon	590.75	715.25	9.10 MB
Tue	570.25	688.50	12.54 MB
Wed	602	721.75	10.97 MB
Thu	577.20	694.20	8.77 MB
Fri	565.80	648	7.61 MB
Sat	534.25	564	3.31 MB
Sun	536.75	572.75	3.89 MB

Figure: Web Visit Aggregates for Weekdays in January 2009

1. Talking about users, the system has been fairly advertised for about 6 months now, and there are a few users visiting my server. The following is the average page hits for each of the weekdays in January of 2009.
2. These are just the raw hit count, they account for around 15-25 unique visitors on the week days and 7-10 users on the weekends

Visits duration		
Number of visits: 414 - Average: 852 s	Number of visits	Percent
0s-30s	236	57 %
30s-2mn	40	9.6 %
2mn-5mn	23	5.5 %
5mn-15mn	16	3.8 %
15mn-30mn	4	0.9 %
30mn-1h	8	1.9 %
1h+	84	20.2 %
Unknown	3	0.7 %

Figure: Web Visit Durations for January 2009

1. Some of the users did not last very long, but also some stayed over an hour. For the users that left quickly they may have been able to quickly find the results, or use the summary table to see that the design they are looking for is not available. Staying too long may mean that they liked the system, and were extensively using it to gather multiple block designs for their work. Further analysis of these times would be needed to figure out the weaknesses and strengths of the web interface.
2. Users from various countries have been hitting the web application.

In conclusion, my project

1. made three large contributions to the field of Combinatorial Designs. Starting with a new specification Ext Rep *v*3, which is thought to be more popular than its predecessor. Design DB hosting over 2.5M pairwise non-isomorphic designs. Also an interface that allows users to search through, find and download block designs.
2. My project also had contributions in the field of computer science, in terms of System Integration, where my system is based on smaller subsystems that were integrated together. Source Code for the project may have benefits to researchers and programmers in the future who may need to apply some of the methods employed in this project. I have also had to evaluate different software packages for the project which I hope to be beneficial to computer scientists in the future.

1. Put the Query Language into the Ext Rep spec
2. Look again at HDBMS, and Document Database CouchDB
3. Study the users, see how the interface can be enhanced
4. When more users start using the system, more machines will have to be utilized to handle the load, and research on the most optimal setup for my application will be required
5. Open Source software is progressing quickly, within the last few months, new versions of Python, Pylons, SA and YAJL have been released and my software should catch up to these new versions to be able to capitalize on the bug fixes and improvements
6. An API would be nice for power users of the system, such that they can programatically search the database without having to resort to the web interface
7. For the system to be a complete central repo, an interface should be added such that new users can easily contribute to