Software Documentation for Web CA

COS 301 Project
2014/05/23
V1.0

for

Christopher David Crossman - crossman.cd@gmail.com
Stephanus Dawid Viljoen - stephanus.daar@gmail.com
Christo Brits - cbrits@live.com

Table of Contents

Table of Figures

1. Vision

There is a need for a web based cellular automata editor and simulator for academics and software professionals to create and share knowledge about cellular automata. The software is required to allow users to create, share, and collaborate cellular automata rules and environments in a 1 dimensional, 2 dimensional, and 3 dimensional world.

2. Background

Currently there are no customizable cellular automata software that can be used by people who have no knowledge of software programming.

All software that make use of cellular automata have a set of predefined rules that can only be changed by the person who developed the software.

Cellular automata can be used by a vast number of academic fields to create multiple solutions to various challenged faced in our daily lives.

3. Architectural Requirements

3.1. Access Channel

Users will be able to access Web CA from any form of personal computer or laptop that can make use of a web browser. Bonus will be to add the functionality to cellular telephones by creating a mobile application (Only if there is time)

3.2. Architectural Constraints

The following Requirements are given by Rishal Hurbans, the team lead software engineer of Entelect and our contact person for the Main COS 301 Project.

AR01: Should consist of a robust REST backend to serve data. Also, a rich JavaScript frontend to process the data for the simulations and user input.
AR02: Must scale any given number of users logging in to the site.
AR03: Must provide secure authentication and authorization.
AR04: Must recover gracefully from failures.
AR05: Must be developed with Java, JavaScript and HTML5.

3.3. Quality

Accessibility:
Users should be able to access the Web CA site from anywhere in the world.
This can be done by providing a server that runs fulltime and that is connected to the Internet to give access to anyone that can access the Internet.

Privacy/Security:
User should be able to feel safe with their personal information that they give when signing up to use Web CA. Their information will also have to be kept safe at all times.
Spring Security provides top of the line authentication and access-control to ensure that user information stays safe.

Scalability:
The system needs to react quickly on input or commands. Performance and reliability should not be dependent on the amount of users on the system.

This can be achieved by creating a robust REST backend that serves data and a rich JavaScript frontend to handle simulations and user input.

Portability:
Users should be able to use any browser that they are comfortable with to access Web CA and all its functionality.
Creating a HTML5 page that is up to standard will ensure that this quality is possible.

Usability:
Users should feel comfortable with Web CA and the way it works. The site should be easy to understand and be visible to the user.
Web CA will be "gamified" to create an appealing user interface that is easy to use, but also fun to use no matter the age of the user.

Persistence:
User data about their Worlds, Cells/States and Rules should always be available to them and never discarded.
Making use of a database to store all information will remedy this.

4. Software Architecture Documentation

4.1. Languages

Java will be used mainly to create the backend of Web CA.
JavaScript, HTML5 and CSS3 will be used to create the frontend of Web CA.

4.2. Architectural Patterns

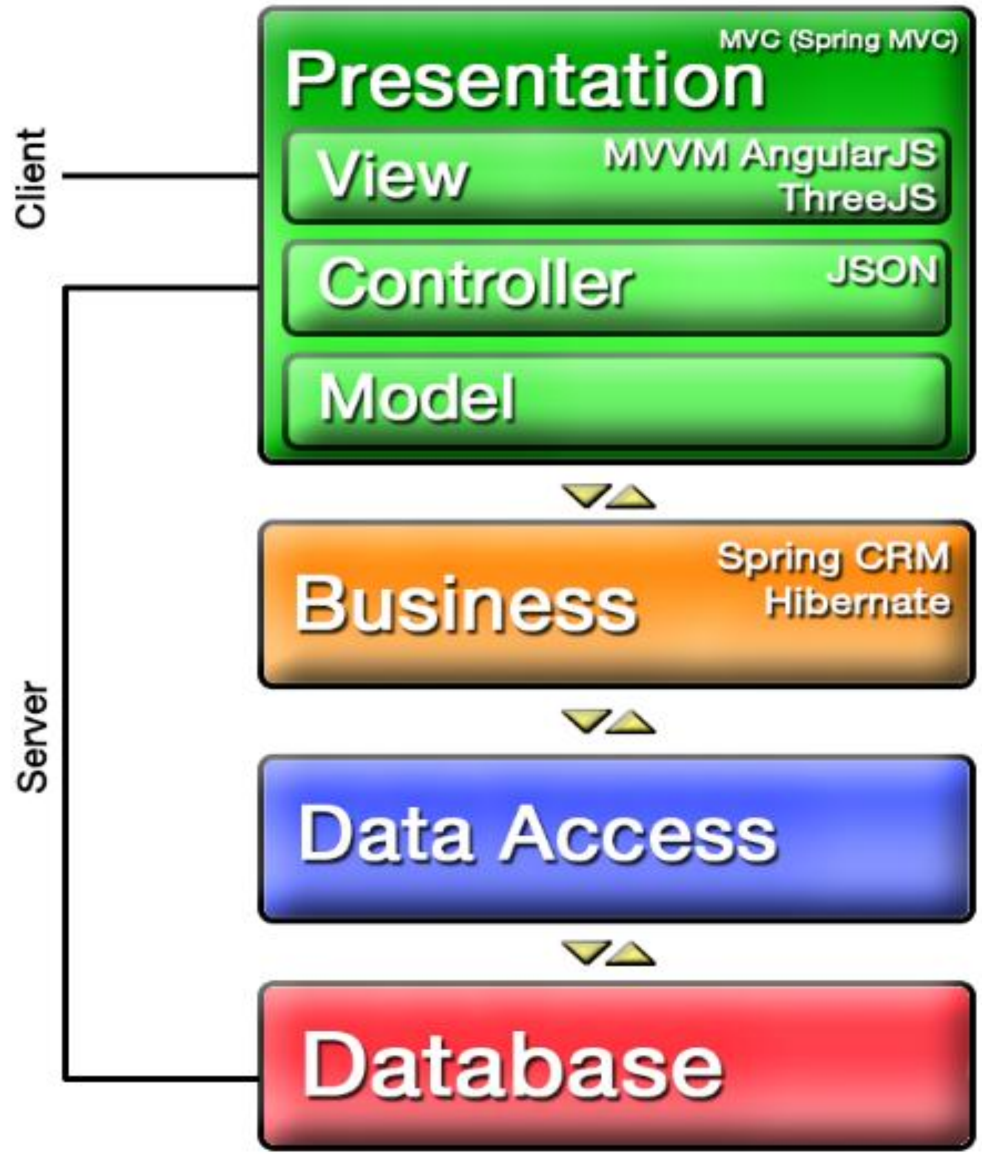

Figure 1: Architectural Patterns

Multi-Tier
Using a multi-tier architecture grants the ability to change the platform Web CA can run on, from a web browser on a personal computer to a mobile application on a cellular telephone.
Separating the different tiers lets multiple workflows be created so that we as a team can independently work on different components of the project without having to wait for other team members to complete their part and don't have to worry about one components letting another component fail.

MVC
The Presentation tier will implement the MVC pattern to help with implementing the user interfaces whilst helping with the representational state transfer of the backend to the frontend and vice versa. This will be the main area where the backend and frontend of the project comes together and scalability will depend greatly on this area.

MVVM
The View inside of the Presentation tier will implement the MVVM pattern asynchronously to ensure that the user experiences a sleek and appealing user interface.

4.3. Technologies

Spring MVC
Very popular Web MVC framework that adds flexibility to one's project. Adding separation of roles, adaptability, customizability and plug-ability to your project.

Spring Security
Powerful and highly customizable authentication and access-control framework. Focusses on providing both authentication and authorization to Java applications.

AngularJS
Open-source web application framework, that assists with creating single-page application that only require HTML, CSS and JavaScript on the client side.

ThreeJS
A lightweight cross-browser JavaScript library/API used to create and display animated 3D computer graphics on a web browser.

JSON
An open standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs.

5. Functional and Non-Functional Requirements

5.1. Functional

The following Requirements are given by Rishal Hurbans, the team lead software engineer of Entelect and our contact person for the Main COS 301 Project.

FR01: Users can register on the site.
FR02: Users can log on to the site.
FR03: Users can create, edit and delete Worlds.
FR04: Users can create, edit and delete Rules for Cells using the defined CA language.
FR05: Users can add and remove Cells in a Worlds and attach one or many Rules to the respective Cells.
FR06: Users can run simulations of the CA World
FR07: A global library of rules and environments should exist for users to use. Also, users should be able to contribute to this library.

5.2. Non-Functional

The following Requirements are given by Rishal Hurbans, the team lead software engineer of Entelect and our contact person for the Main COS 301 Project.

NF01: Tiered architecture for the backend.
NF02: Asynchronous MVVM architecture for the frontend.
NF03: Users must be able to add additional plugins to the backend and frontend.

5.3. Scope

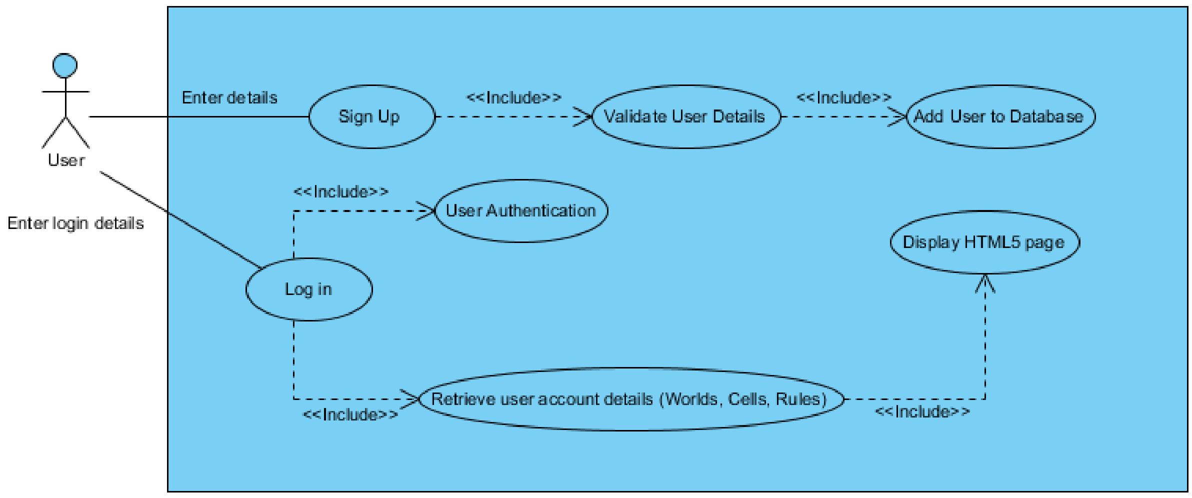

Figure 3: User Signup and Login

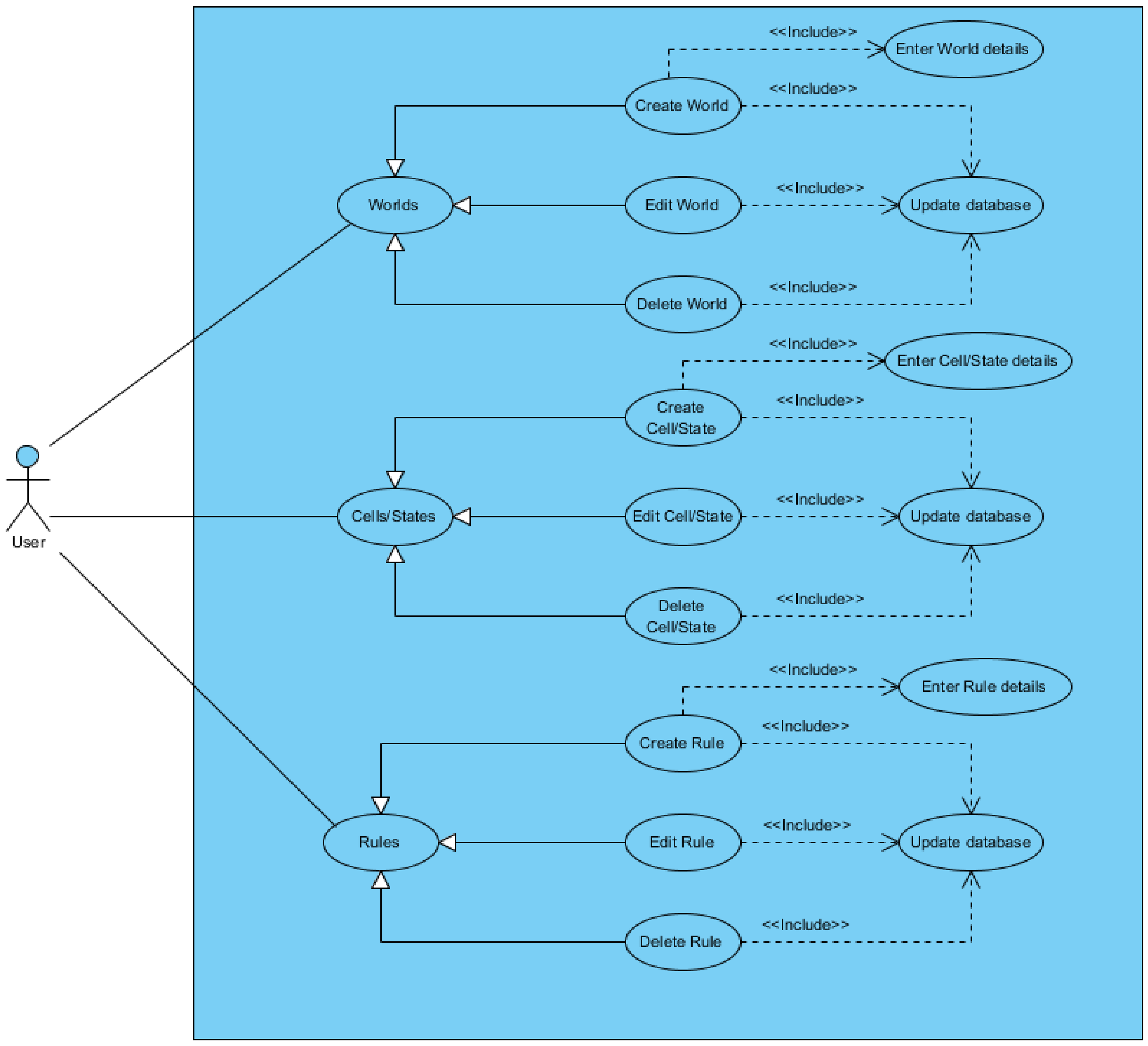

Figure 2: Managing Worlds, Cells/States and Rules

5.4. Process Specifications

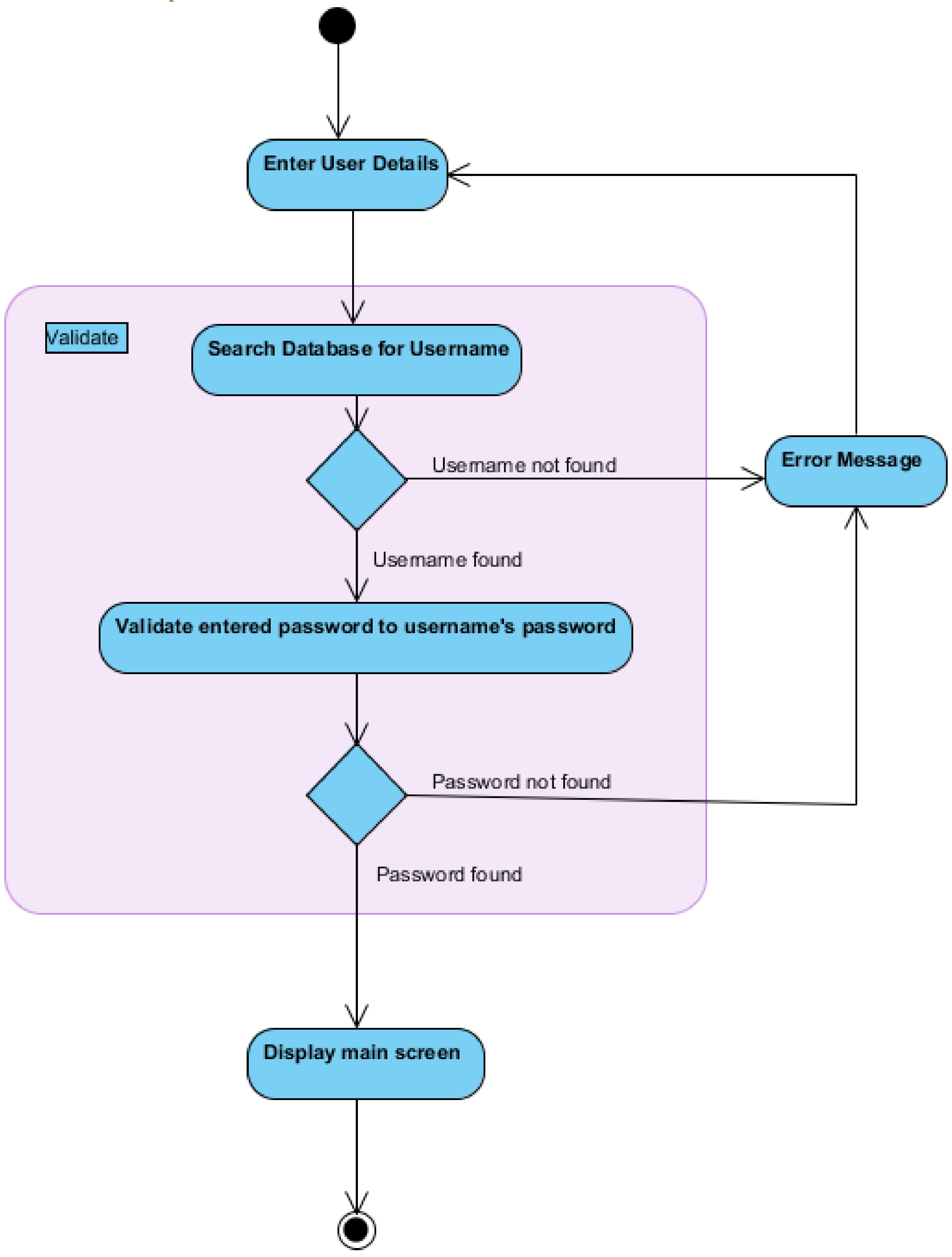

Figure 4: User Login Activity

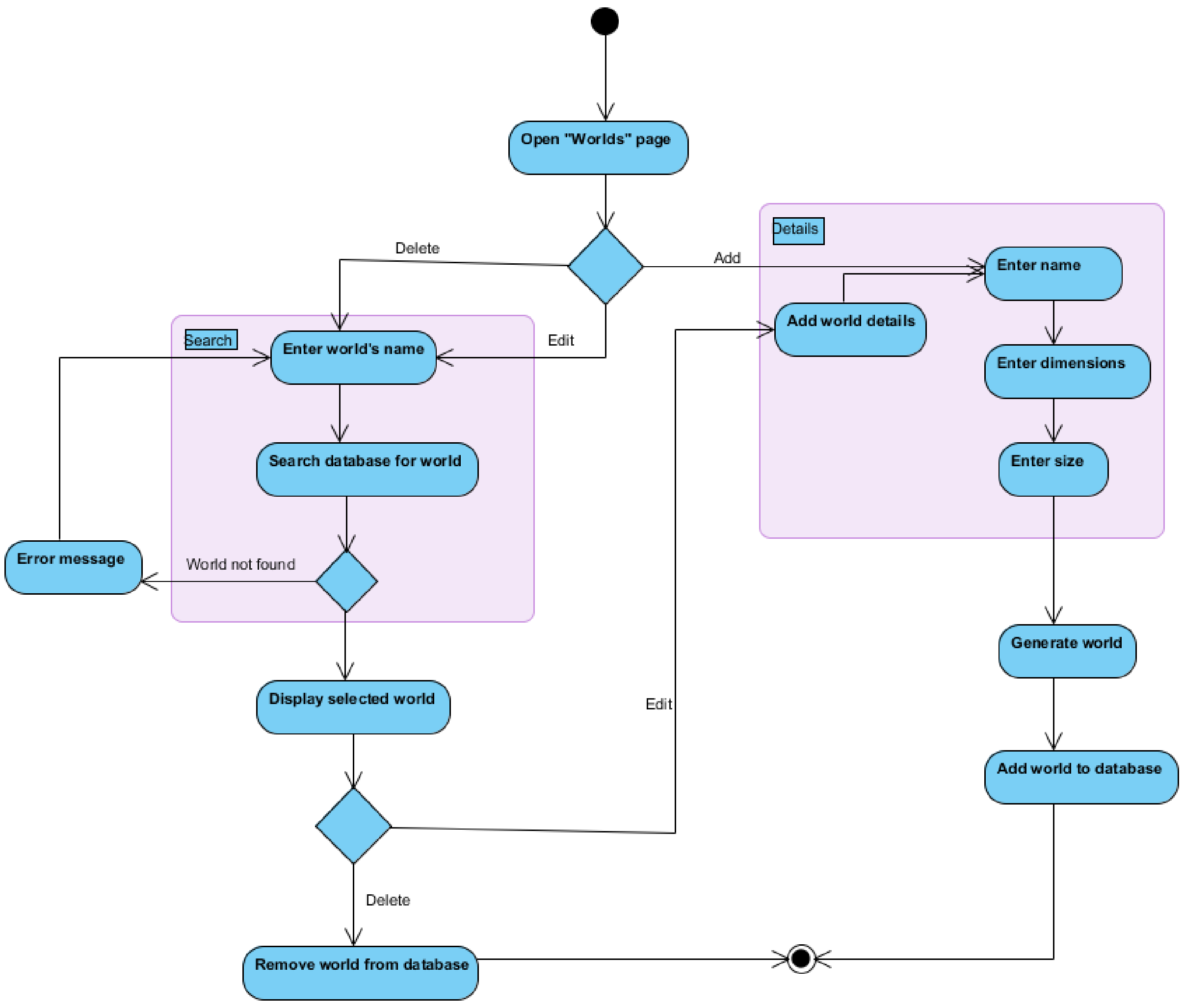

Figure 5: Managing World Activity

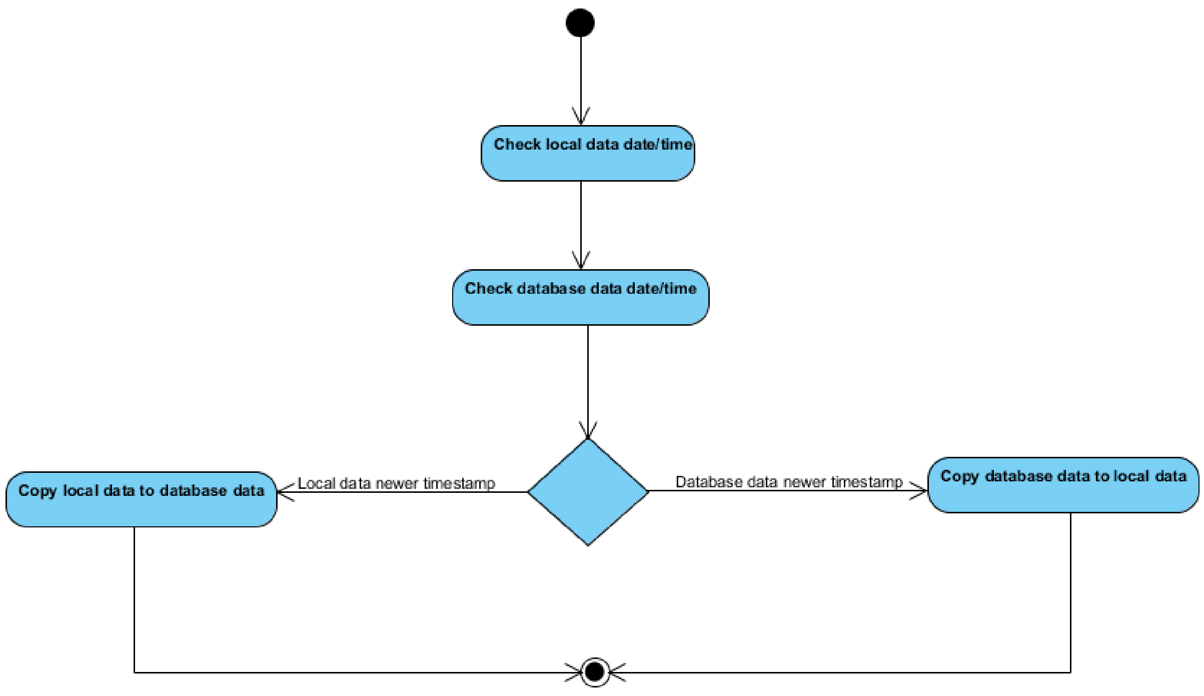
Check local data date/time
Check database data date/time
Copy local data to database data
Local data newer timestamp
Database data newer timestamp
Copy database data to local data

Figure 6: Sync Database

5.5. Use Case Prioritization

Critical Use Cases are the main cases that the system is made up of namely, Logging In, Creating Worlds, Creating Cells/States and Creating Rules, also the case where the user can run a simulation of their World. Without these cases the system will have limited to no functionality which will lead to a system that is not required by anyone

Important Use Cases are the cases that improves the critical use cases and introduces a wider variety of functionality. These cases are the Edit and Delete of the Worlds, Cells/States and Rules.

Nice-To-Have Use Cases make the system more user friendly and provides a better user experience. This case is the Share World use case, this is where users can browse each other's Worlds, Cells/States and Rules and copy it and use it for themselves.

5.6. Use Case Contracts

Pre-Conditions:
Create, Edit and Delete is regarding Worlds, Cells/States and Rules
Signup – User information does not exist in database
Login – User information is stored in the database
Create – Object does not exist in database
Edit – Object already exists in database
Delete – Object already exists in database
Share – Object already exists in database of owner of the object and does not exist in current user

Post-Conditions:
Create, Edit and Delete is regarding Worlds, Cells/States and Rules
Signup – User information is added to database
Login – User information has been validated
Create – Object is added to database and users are able to use it
Edit – Object details are altered in database and users are able to use it
Delete – Object is removed from database
Share – Object is added to current user's database and owner of object is notified/rated